



Red Hat Certificate System 8.0 Install Guide

Installing Red Hat Certificate System 8.0
Edition 8.0.13

Landmann

Red Hat Certificate System 8.0 Install Guide

Installing Red Hat Certificate System 8.0 Edition 8.0.13

Landmann
rlandmann@redhat.com

Legal Notice

Copyright © 2009 Red Hat, Inc..

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, MetaMatrix, Fedora, the Infinity Logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This guide explains how to install and configure Red Hat Certificate System subsystems, as well as covering some basic administrative tasks and advanced installation techniques. This guide also lists the supported platforms and dependencies for Red Hat Certificate System; for other information, see the Release Notes.

Table of Contents

| | |
|---------------------------------------------------------------------------------------|-----------|
| About This Guide | 5 |
| 1. Examples and Formatting | 5 |
| 1.1. Formatting for Examples and Commands | 5 |
| 1.2. Tool Locations | 5 |
| 1.3. Guide Formatting | 5 |
| 2. Additional Reading | 6 |
| 3. Giving Feedback | 7 |
| 4. Document History | 7 |
| Chapter 1. Overview of Certificate System Subsystems | 9 |
| 1.1. Subsystems for Managing Certificates | 9 |
| 1.1.1. Certificate Manager | 10 |
| 1.1.2. Registration Authority | 10 |
| 1.1.3. Data Recovery Manager | 10 |
| 1.1.4. Online Certificate Status Manager | 11 |
| 1.2. Subsystems for Managing Tokens | 11 |
| 1.2.1. Token Processing System | 12 |
| 1.2.2. Token Key Service | 12 |
| 1.2.3. Enterprise Security Client | 12 |
| 1.3. Planning the Installation | 12 |
| Chapter 2. Prerequisites Before Installing Certificate System | 14 |
| 2.1. Supported Platforms, Hardware, and Programs | 14 |
| 2.1.1. Supported Platforms | 14 |
| 2.1.2. Supported Web Browsers | 14 |
| 2.1.3. Supported Smart Cards | 15 |
| 2.1.4. Supported HSM | 15 |
| 2.1.5. Supported Charactersets | 15 |
| 2.2. Required Programs, Dependencies, and Configuration | 16 |
| 2.2.1. Java Development Kit (JDK) | 16 |
| 2.2.2. Apache | 16 |
| 2.2.3. Red Hat Directory Server | 17 |
| 2.2.4. Additional Packages | 17 |
| 2.2.5. Firewall Configuration and iptables | 18 |
| 2.2.6. SELinux Settings | 18 |
| 2.3. Packages Installed on Red Hat Enterprise Linux | 18 |
| 2.4. Required Information for Subsystem Configuration | 19 |
| 2.5. Setting up Tokens for Storing Certificate System Subsystem Keys and Certificates | 21 |
| 2.5.1. Types of Hardware Tokens | 21 |
| 2.5.1.1. Internal Tokens | 21 |
| 2.5.1.2. External Tokens | 21 |
| 2.5.1.3. Hardware Cryptographic Accelerators | 21 |
| 2.5.2. Using Hardware Security Modules with Subsystems | 22 |
| 2.5.2.1. Adding or Managing the HSM Entry for a Subsystem | 22 |
| 2.5.2.2. Using Chrysalis LunaSA HSM | 22 |
| 2.5.2.3. Installing External Tokens and Unsupported HSM | 24 |
| 2.5.2.4. Setting up SELinux on nCiper netHSM 2000 | 25 |
| 2.5.3. Viewing Tokens | 25 |
| 2.5.4. Detecting Tokens | 25 |
| Chapter 3. Installation and Configuration | 26 |
| 3.1. Overview of Installation | 26 |

| | |
|------------------------------------------------------------------|------------|
| 3.2. Installing the Certificate System Packages | 27 |
| 3.2.1. Installing through yum | 28 |
| 3.2.2. Installing from an ISO Image | 29 |
| 3.3. Configuring a CA | 29 |
| 3.4. Configuring an RA | 37 |
| 3.5. Configuring a DRM, OCSP, or TKS | 42 |
| 3.6. Configuring a TPS | 47 |
| Chapter 4. Additional Installation Options | 55 |
| 4.1. Requesting Subsystem Certificates from an External CA | 55 |
| 4.2. Installing a CA with ECC Enabled | 57 |
| 4.2.1. Loading a Third-Party ECC Module | 57 |
| 4.2.2. Loading the Certicom ECC Module | 58 |
| 4.3. Changing the Hashing Algorithm Used for Subsystem Keys | 62 |
| 4.4. Enabling IPv6 for a Subsystem | 63 |
| 4.5. Configuring Separate RA Instances | 64 |
| Chapter 5. Creating Additional Subsystem Instances | 68 |
| 5.1. About pkicreate | 68 |
| 5.2. Running pkicreate for a Single SSL Port | 71 |
| 5.3. Running pkicreate with Port Separation | 71 |
| Chapter 6. Cloning Subsystems | 73 |
| 6.1. About Cloning | 73 |
| 6.1.1. Cloning for CAs | 74 |
| 6.1.2. Cloning for DRMs | 75 |
| 6.1.3. Cloning for Other Subsystems | 75 |
| 6.1.4. Cloning and Key Stores | 75 |
| 6.1.5. LDAP Considerations | 76 |
| 6.2. Exporting Keys from a Software Database | 76 |
| 6.3. Cloning a CA | 76 |
| 6.4. Cloning OCSP Subsystems | 79 |
| 6.5. Cloning DRM and TKS Subsystems | 82 |
| 6.6. Converting Masters and Clones | 85 |
| 6.6.1. Converting CA Clones and Masters | 85 |
| 6.6.2. Converting OCSP Clones | 86 |
| 6.7. Updating CA Clones | 87 |
| Chapter 7. Silent Configuration | 88 |
| 7.1. About pkisilent | 88 |
| 7.2. Silently Configuring Subsystem | 93 |
| 7.3. Cloning a Subsystem Silently | 96 |
| 7.4. Performing Silent Configuration Using an External CA | 97 |
| Chapter 8. Updating and Removing Subsystem Packages | 100 |
| 8.1. Updating Certificate System Packages | 100 |
| 8.2. Uninstalling Certificate System Subsystems | 101 |
| 8.2.1. Removing a Subsystem Instance | 101 |
| 8.2.2. Removing Certificate System Subsystem Packages | 101 |
| Chapter 9. Using Certificate System | 103 |
| 9.1. Starting the Certificate System Console | 103 |
| 9.2. Starting, Stopping, and Restarting an Instance | 103 |
| 9.3. Starting the Subsystem Automatically | 103 |
| 9.4. Finding the Subsystem Web Services Pages | 105 |
| 9.5. Default File and Directory Locations for Certificate System | 109 |
| 9.5.1. Default CA Instance Information | 109 |

| | |
|-----------------------------------------------------------|------------|
| 9.5.2. Default RA Instance Information | 110 |
| 9.5.3. Default DRM Instance Information | 111 |
| 9.5.4. Default OCSP Instance Information | 112 |
| 9.5.5. Default TKS Instance Information | 113 |
| 9.5.6. Default TPS Instance Information | 114 |
| 9.5.7. Shared Certificate System Subsystem File Locations | 115 |
| Index | 117 |
| A | 117 |
| C | 117 |
| E | 117 |
| H | 117 |
| I | 117 |
| K | 118 |
| P | 118 |
| S | 118 |
| T | 118 |

About This Guide

This guide explains how to install and configure Red Hat Certificate System subsystems, as well as covering some basic administrative tasks and advanced installation techniques. This guide also lists the supported platforms and dependencies for Red Hat Certificate System; for other information, see the Release Notes.

The *Certificate System Deployment Guide* explains different usage scenarios, and it is a good idea to understand the major concepts in that guide before installing the Certificate System subsystems. The *Certificate System Administrator's Guide* covers all of the administrative tasks of Red Hat Certificate System, such as configuring logging, setting up certificate and CRL publishing, requesting and issuing certificates, and building CRLs.

Certificate System agents should refer to the *Certificate System Agent's Guide* for information on performing agent tasks, such as handling certificate requests and revoking certificates. For information on using Certificate System to manage smart cards and security tokens, see *Managing Smart Cards with the Enterprise Security Client*.

1. Examples and Formatting

1.1. Formatting for Examples and Commands

All of the examples for Red Hat Certificate System commands, file locations, and other usage are given for Red Hat Enterprise Linux 5 (32-bit) systems. Be certain to use the appropriate commands and files for your platform.

Example 1. Example Command

To start the Red Hat Certificate System:

```
service pki-ca start
```

1.2. Tool Locations

All of the tools for Red Hat Certificate System are located in the `/usr/bin` directory. These tools can be run from any location without specifying the tool location.

1.3. Guide Formatting

Certain words are represented in different fonts, styles, and weights. Different character formatting is used to indicate the function or purpose of the phrase being highlighted.

| Formatting Style | Purpose |
|----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| Monospace font | Monospace is used for commands, package names, files and directory paths, and any text displayed in a prompt. |
| <div>Monospace with a background</div> | This type of formatting is used for anything entered or returned in a command prompt. |
| <i>Italicized text</i> | Any text which is italicized is a variable, such as <i>instance_name</i> or <i>hostname</i> . Occasionally, this is also used |

Bolded text

to emphasize a new term or other phrase.

Most phrases which are in bold are application names, such as **Cygwin**, or are fields or options in a user interface, such as a **User Name Here**: field or **Save** button.

Other formatting styles draw attention to important text.

**NOTE**

A note provides additional information that can help illustrate the behavior of the system or provide more detail for a specific issue.

**IMPORTANT**

Important information is necessary, but possibly unexpected, such as a configuration change that will not persist after a reboot.

**WARNING**

A warning indicates potential data loss, as may happen when tuning hardware for maximum performance.

2. Additional Reading

The documentation for Certificate System includes the following guides:

- ▶ [Certificate System Deployment Guide](#) describes basic PKI concepts and gives an overview of the planning process for setting up Certificate System.
This manual is intended for Certificate System administrators.
- ▶ [Certificate System Installation Guide](#) covers the installation process for all Certificate System subsystems.
This manual is intended for Certificate System administrators.
- ▶ [Certificate System Administrator's Guide](#) explains all administrative functions for the Certificate System. Administrators maintain the subsystems themselves, so this manual details backend configuration for certificate profiles, publishing, and issuing certificates and CRLs. It also covers managing subsystem settings like port numbers, users, and subsystem certificates.
This manual is intended for Certificate System administrators.
- ▶ [Certificate System Agent's Guide](#) describes how agents — users responsible for processing certificate requests and managing other aspects of certificate management — can use the Certificate System subsystems web services pages to process certificate requests, key recovery, OCSP requests and CRLs, and other functions.
This manual is intended for Certificate System agents.
- ▶ [Managing Smart Cards with the Enterprise Security Client](#) explains how to install, configure, and use the Enterprise Security Client, the user client application for managing smart cards, user certificates, and user keys.
This manual is intended for Certificate System administrators, agents, privileged users (such as

security officers), and regular end users.

- [Using End User Services](#) is a quick overview of the end-user services in Certificate System, a simple way for users to learn how to access Certificate System services.

This manual is intended for regular end users.

- [Certificate System Command-Line Tools Guide](#) covers the command-line scripts supplied with Red Hat Certificate System.

This manual is intended for Certificate System administrators.

- [Certificate System Migration Guide](#) covers version-specific procedures for migrating from older versions of Certificate System to Red Hat Certificate System 8.0.

This manual is intended for Certificate System administrators.

- [Release Notes](#) contains important information on new features, fixed bugs, known issues and workarounds, and other important deployment information for Red Hat Certificate System 8.0.

All of the latest information about Red Hat Certificate System and both current and archived documentation is available at <http://www.redhat.com/docs/manuals/cert-system/>.

3. Giving Feedback

If there is any error in this *Installation Guide* or there is any way to improve the documentation, please let us know. Bugs can be filed against the documentation for Red Hat Certificate System through Bugzilla, <http://bugzilla.redhat.com/bugzilla>. Make the bug report as specific as possible, so we can be more effective in correcting any issues:

- Select the Red Hat Certificate System product.
- Set the component to **Doc - quick install guide**.
- Set the version number to 8.0.
- For errors, give the page number (for the PDF) or URL (for the HTML), and give a succinct description of the problem, such as incorrect procedure or typo.
For enhancements, put in what information needs to be added and why.
- Give a clear title for the bug. For example, "**Incorrect command example for setup script options**" is better than "**Bad example**".

We appreciate receiving any feedback — requests for new sections, corrections, improvements, enhancements, even new ways of delivering the documentation or new styles of docs. You are welcome to contact Red Hat Content Services directly at docs@redhat.com.

4. Document History

| Revision 8.0.12 | September 16, 2010 | Ella Deon Lackey |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|------------------|
| Adding extra step for setting SELinux context on files used for cloning, per Bugzilla #627054. Adding required additional parameters for the second step in using pkisilent with an external CA and the key_algorithm, per Bugzilla #631003. | | |
| Revision 8.0.11 | June 1, 2010 | Ella Deon Lackey |
| Adding back Mac support, on a new platform (10.5.8), per Errata RHBA-2010:0448. | | |
| Revision 8.0.10 | March 25, 2010 | Ella Deon Lackey |
| Adding information on new end-entities client authentication port for the CA, related to the MitM resolution in Errata RHBA-2010:0169. | | |

| | | |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|-------------------------|
| Revision 8.0.9 | February 18, 2010 | Ella Deon Lackey |
| Changing LunaSA setup, per Bugzilla 537529. Correcting the cs_port definition, per Bugzilla 533303. | | |
| Revision 8.0.8 | December 21, 2009 | Ella Deon Lackey |
| Updating platform support to include 64-bit Windows platforms, per Errata RHBA-2009:1687. | | |
| Revision 8.0.7 | November 25, 2009 | Ella Deon Lackey |
| Updating the pkisilent documentation and expanding the CA configuration section to add ability to specify the CA signing algorithm at installation, per Errata RHBA-2009:1602. | | |
| Revision 8.0.6 | November 12, 2009 | Ella Deon Lackey |
| Updating the Certicom ECC configuration per comment #3 in Bugzilla 507428. Correcting the 'rpm -qi' example in checking prerequisites, per QE feedback. Adding note about (no) 64-bit support for ESC. | | |
| Revision 8.0.5 | November 3, 2009 | Ella Deon Lackey |
| Adding information on setting the NSS_USE_DECODED_CKA_EC_POINT environment variable for the console and clarifying the security database argument in the modutil step for configuring the Certicom ECC module. | | |
| Revision 8.0.4 | August 28, 2009 | Ella Deon Lackey |
| Removing the draft watermarks. Tech reviews for chapter 2 (prerequisites) and chapter 9 (basic usage), per Bugzilla #510578 and #510587. | | |
| Revision 8.0.3 | August 11, 2009 | Ella Deon Lackey |
| Tech reviews for chapter 8, per Bugzilla #510586. Adding extra bullet point on using unique certificate nicknames for HSMs, related to Bugzilla #510987. | | |
| Revision 8.0.2 | August 3, 2009 | Ella Deon Lackey |
| Tech reviews for chapter 4, per Bugzilla #510580. | | |
| Revision 8.0.1 | July 26, 2009 | Ella Deon Lackey |
| Tech reviews for chapters 1, 3, 5, 6, and 7, per Bugzilla #510577, #510579, #510581, #510582, #510585. Adding a section for configuring netHSM to work with SELinux, per Bugzilla #513312, with cross references in the instance configuration sections. | | |
| Revision 8.0.0 | July 22, 2009 | Ella Deon Lackey |
| Initial draft for Certificate System 8.0 <i>Installation Guide</i> . | | |

Chapter 1. Overview of Certificate System Subsystems

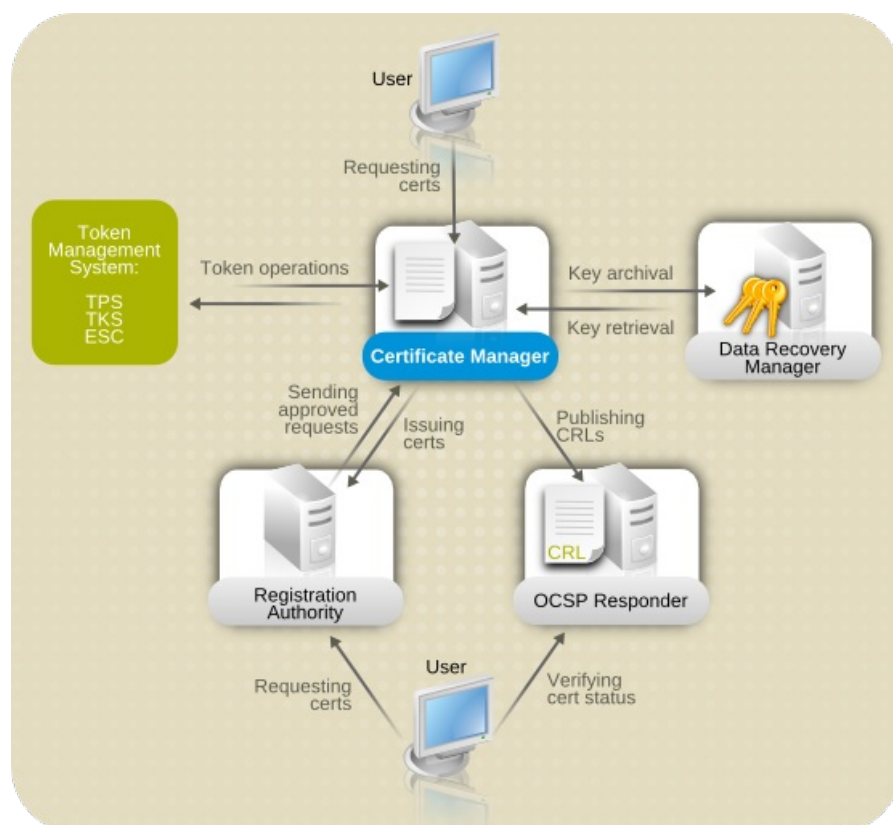
Red Hat Certificate System is a highly configurable set of components which create and manage certificates and keys at every point of the certificate lifecycle. Certificate System is based on open standards so that it effectively creates a scalable, customizable, and robust public-key infrastructure (PKI).

Certificate System has subsystems which discretely handle different PKI management functions. These subsystems interact in flexibly established ways to manage certificates and to manage tokens.

1.1. Subsystems for Managing Certificates

Up to four Certificate System subsystems work together to manage certificates:

- Certificate Manager, a certificate authority (CA) which issues, renews, and revokes certificates and publishes certificate revocation lists (CRLs)
- Data Recovery Manager (DRM), which archives and recovers keys
- Online Certificate Status Manager (OCSP), which processes requests for certificates' revocation status
- Registration Authority (RA), which validates and approves certificate requests locally and then forwards approval to the CA to issue the certificates



The core of the Certificate System is the Certificate Manager. This is the only required subsystem and handles the actual certificate management tasks. The other subsystems can be added for additional functionality (the DRM) or to move the load for some common operations off of the CA, such as using an OCSP for status requests and an RA to process certificate requests.

The CA, RA, DRM, and OCSP are the subsystems used to manage certificates, keys, and CRLs, through every step of the cycle of a certificate:

1. Generating a certificate request (CA or RA)
2. Submitting the request to a CA (CA or RA)
3. Generating key pairs (CA)
4. Storing the key pairs (DRM)
5. Issuing the certificate (CA)
6. Recovering the keys if the certificate is lost (DRM)
7. Revoking the certificate (CA)
8. Checking whether the certificate is revoked or active when an entity tries to use the certificate for authentication (OCSP)

All of the subsystems (CA, RA, DRM, and OCSP, as well as the token subsystems) are organized together in *security domains*. Security domains define communication between subsystems, and this makes the Certificate System very efficient. For example, if a user needs to archive keys, the request can be processed by the first available DRM in the domain, automatically. The available DRMs in the domain are automatically updated every time a new DRM is configured.

1.1.1. Certificate Manager

The Certificate Manager subsystem is a certificate authority. It issues, renews, revokes, and publishes a wide variety of certificates: for servers, for users, for routers, for other subsystems, and for file or object signing. The Certificate Manager also compiles and publishes CRLs.

Certificate Managers can be structured in series (*hierarchy*), so that one Certificate Manager sets policies and issues signing certificates to a *subordinate CA*. The highest Certificate Manager in the chain is a *root CA*.

A special kind of certificate is used by CAs to sign certificates they issue, sort of like a stamp or seal. This is called a *CA signing certificate*. A subordinate CA is issued a CA signing certificate by a CA higher in the hierarchy, and the parameters of the CA signing certificate are set by the superior CA. A CA which issues its own signing certificate has a *self-signed certificate*. There are benefits to having a self-signed CA certificate for your root CA, as well as some benefits to having the certificate signed by a third-party CA.

Additionally, a Certificate Manager is always the subsystem which works as the *registry* for the security domain. The very first Certificate Manager configured must create a security domain, but every Certificate Manager configured after has the option of joining an existing security domain rather than creating a new one. The configuration of your PKI deployment determines whether you need multiple security domains; for more information, see the *Red Hat Certificate System Deployment Guide*.

1.1.2. Registration Authority

The Registration Authority subsystem handles certain certificate issuing tasks locally, such as generating and submitting certificate requests. This effectively makes the RA a load-balancer for the CA; a local RA can receive and verify the legitimacy of a certificate request (*authenticate* it) and then forward valid requests to the CA to issue the certificate. Certificates can also be retrieved through the RA and the status of the request can be checked through the RA, both of which lower demand on the CA.

The RA is normally set up outside of the firewall, and the CA is set up behind the firewall so that requests can be submitted to Certificate System externally, while the CA is protected.

The RA accepts requests for a smaller number of certificate types than the CA, including user, server, and router certificates.

1.1.3. Data Recovery Manager

The Data Recovery Manager (DRM) is a *key recovery authority*, which means it works with the Certificate Manager when a certificate is issued and stores private encryption keys. Those private keys can be restored (in a PKCS #12 file) if a certificate is lost.

NOTE

The DRM only archives encryption keys, not signing keys, because that compromises the non-repudiation properties of signing keys. Non-repudiation means that a user cannot deny having performed some action, such as sending an encrypted email, because they are the only possessor of that key.

1.1.4. Online Certificate Status Manager

The Online Certificate Status Manager is an OCSP service, external to the Certificate Manager. Although the Certificate Manager is configured initially with an internal OCSP service, an external OCSP responder allows the OCSP subsystem to be outside the firewall and accessible externally, while keeping the Certificate Manager behind the firewall. Like the RA, the OCSP acts as a load-balancer for requests to the Certificate Manager.

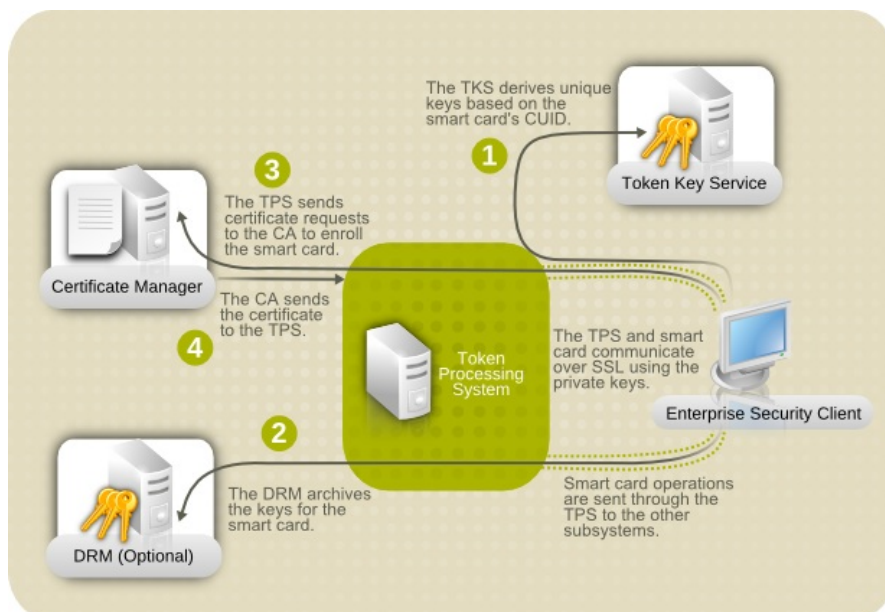
The Online Certificate Status Manager verifies the status of a certificate by checking a certificate revocation list, published by the Certificate Manager, to see if the specified certificate has been revoked. More than one Certificate Manager can publish CRLs to a single OCSP.

1.2. Subsystems for Managing Tokens

Two subsystems are required to manage tokens:

- Token Processing System (TPS), which accepts operations from a token and forwards them to the CA (for processing certificate requests, renewal, issuing, and revocation) and to the DRM (to archive or restore keys)
- Token Key Service (TKS), which generates master keys and symmetric keys for the TPS to use when communicating with other subsystems

A third application, the Enterprise Security Client, is the interface between the user and the TPS.



1.2.1. Token Processing System

The Token Processing System (TPS) is the conduit between the user-centered Enterprise Security Client, which interacts with the tokens, and the Certificate System backend subsystems, such as the Certificate Manager. The TPS is required in order to manage smart cards.

The TPS communicates with the CA and DRM for processing token operations. The TPS also communicates with the TKS to derive token-specific secret keys.

1.2.2. Token Key Service

The Token Key Service (TKS) uses a master key to derive specific, separate keys for every smart card. The TPS uses these secret keys to communicate with each smart card securely, since all communication between the TPS and the smart card is encrypted.

The only Certificate System subsystem which the TKS interacts with is the TPS.

1.2.3. Enterprise Security Client

The Enterprise Security Client is not a subsystem since it does not perform any operations with certificates, keys, or tokens. The Enterprise Security Client, as the name implies, is a user interface which allows people to manage certificates on smart cards very easily. The Enterprise Security Client sends all token operations, such as certificate requests, to the TPS, which then sends them to the CA.

1.3. Planning the Installation

Before beginning to install and configure the Certificate System subsystems, determine what the organization of the PKI is.

Q: What types of subsystems do you need to install?

A: This depends on the kind of functionality you need and the load you expect to have. There are several different kinds of subsystems for managing certificates ([Section 1.1, “Subsystems for Managing Certificates”](#)) and for managing tokens ([Section 1.2, “Subsystems for Managing Tokens”](#)).

Q: How many subsystems do you need to install?

A: This depends very much on the expected load and also on geographical or departmental divisions. Subsystems can be cloned, meaning they essentially are clustered, operating as a single unit, which is good for load balancing and high availability. Additionally, security domains create trusted relationships between subsystems, allowing them to work together to find available subsystems to respond to immediate needs. Multiple security domains can be used in a single PKI, with multiple instances of any kind of subsystem.

Q: Will the subsystem certificates and keys be stored on the internal software token in Certificate System or on an external hardware token?

A: Certificate System supports two hardware security modules (HSM): nCipher netHSM 2000 and Safenet LunaSA. Using a hardware token can require additional setup and configuration before installing the subsystems, but it also adds another layer of security.

Q: What machines should the subsystem be installed on?

A: This depends on the network design. The RA and OCSP subsystems are specifically designed to operate outside a firewall for user convenience, while the CA, DRM, and TPS should all be secured behind a firewall.

Q: To what security domain should a subsystem instance be added?

A: Because the subsystems within a security domain have trusted relationships with each other, it is important what domain a subsystem joins. Security domains can have different certificate issuing policies, different kinds of subsystems within them, or a different Directory Server database. Map out where (both on the physical machine and in relation to each other) each subsystem belongs, and assign it to the security domain accordingly.

Q: Should a subsystem be cloned?

A: Cloned subsystems work together, essentially as a single instance. This can be good for high demand systems, failover, or load balancing, but it can become difficult to maintain. For example, cloned CAs have serial number ranges for the certificates they issue, and a clone could hit the end of its range.

Q: Should the Certificate Manager be a self-signed root CA or a subordinate CA?

A: A Certificate Manager can be configured as either a root CA or a subordinate CA. The difference between a root CA and a subordinate CA is who signs the CA signing certificate. A root CA signs its own certificate. A subordinate CA has another CA (either internal or external) sign its certificate.

A self-signing root CA issues and signs its own CA signing certificate. This allows the CA to set its own configuration rules, like validity periods and the number of allowed subordinate CAs.

A subordinate CA has its certificates issued by a public CA or another Certificate System root CA. This CA is *subordinate* to the other CA's rules about its certificate settings and how the certificate can be used, such as the kinds of certificates that it can issue, the extensions that it is allowed to include in certificates, and the levels of subordinate CAs the subordinate CA can create.

One option is to have the Certificate manager subordinate to a public CA. This can be very restrictive, since it introduces the restrictions that public CAs place on the kinds of certificates the subordinate CA can issue and the nature of the certificate chain. On the other hand, one benefit of chaining to a public CA is that the third party is responsible for submitting the root CA certificate to a web browser or other client software, which is a major advantage for certificates that are accessed by different companies with browsers that cannot be controlled by the administrator.

The other option is make the CA subordinate to a Certificate System CA. Setting up a Certificate System CA as the root CA means that the Certificate System administrator has control over all subordinate CAs by setting policies that control the contents of the CA signing certificates issued.

It is easiest to make the first CA installed a self-signed root, so that it is not necessary to apply to a third party and wait for the certificate to be issued. Make sure that you determine how many root CAs to have and where both root and subordinate CAs will be located.

Chapter 2. Prerequisites Before Installing Certificate System

Before installing the Red Hat Certificate System subsystems, check out the requirements and dependencies for the specific platform, as well as looking at the installed packages.

2.1. Supported Platforms, Hardware, and Programs

2.1.1. Supported Platforms

The Certificate System subsystems (CA, RA, DRM, OCSP, TKS, and TPS) are supported on the following platforms:

- Red Hat Enterprise Linux 5.3 (x86, 32-bit)
- Red Hat Enterprise Linux 5.3 (x86_64, 64-bit)

The Enterprise Security Client, which manages smart cards for end users, is supported on the following platforms:

- Red Hat Enterprise Linux 5.3 (x86, 32-bit)
- Red Hat Enterprise Linux 5.3 (x86_64, 64-bit)
- Microsoft Windows Vista 32-bit
- Microsoft Windows Vista 64-bit
- Microsoft Windows XP 32-bit
- Microsoft Windows XP 64-bit
- Apple Mac OS X 10.5.x (Leopard)

2.1.2. Supported Web Browsers

The services pages for the subsystems require a web browser that supports SSL. It is strongly recommended that users such as agents or administrators use Mozilla Firefox to access the agent services pages. Regular users should use Mozilla Firefox or Microsoft Internet Explorer.



NOTE

The only browser that is fully-supported for the HTML-based instance configuration wizard is Mozilla Firefox.

Table 2.1. Supported Web Browsers by Platform

| Platform | Agent Services | End User Pages |
|--------------------------|------------------------------------------|-----------------------------------------------|
| Red Hat Enterprise Linux | Firefox 3.x | Firefox 3.x |
| Windows Vista | Firefox 2.x | Firefox 2.x Internet Explorer 7 and higher |
| Windows XP | Firefox 2.x | Firefox 2.x Internet Explorer 6 and higher |
| Mac OS 10.5.x | Agent services are not supported for Mac | Firefox 2.x |

2.1.3. Supported Smart Cards

The Enterprise Security Client supports Global Platform 2.01-compliant smart cards and JavaCard 2.1 or higher.

The Certificate System subsystems have been tested using the following tokens:

- Gemalto TOP IM FIPS CY2 64K token, both as a smart card and GemPCKey USB form factor key
- Gemalto Cyberflex e-gate 32K token
- Safenet 330J Java smart card

Smart card testing was conducted using the SCM SCR331 CCID reader.

The only card manager applet supported with Certificate System is the CoolKey applet which ships with Red Hat Enterprise Linux 5.3.

2.1.4. Supported HSM

Red Hat Certificate System supports two hardware security modules (HSM), nCipher netHSM 2000 and Chrysalis-IT LunaSA.

| HSM | Firmware | Appliance Software | Client Software |
|------------------------------|----------|--------------------|-----------------|
| Safenet Chrysalis-ITS LunaSA | 4.5.2 | 3.2.4 | 3.2.4 |
| nCipher netHSM 2000 | 2.33.60 | 11.10 | |

2.1.5. Supported Charactersets

Red Hat Certificate System fully supports UTF-8 characters in the CA end users forms for specific fields. This means that end users can submit certificate requests with UTF-8 characters in those fields and can search for and retrieve certificates and CRLs in the CA and retrieve keys in the DRM when using those field values as the search parameters.

Four fields fully-support UTF-8 characters:

- Common name (used in the subject name of the certificate)
- Organizational unit (used in the subject name of the certificate)

- Requester name
- Additional notes (comments appended by the agent to the certificate)

**NOTE**

This support does not include supporting internationalized domain names, like in email addresses.

2.2. Required Programs, Dependencies, and Configuration

To install any Red Hat Certificate System subsystems on Red Hat Enterprise Linux, three programs are required: OpenJDK, Apache or Tomcat (depending on the subsystem), and Red Hat Directory Server. All other required packages should be present as part of the base Red Hat Enterprise Linux operating system packages.

2.2.1. Java Development Kit (JDK)

Certificate System requires OpenJDK 1.6.0. On Red Hat Enterprise Linux systems, this must be installed separately. The OpenJDK can be installed by using **yum** or by downloading the packages directly from <http://openjdk.java.net/install/>. For example:

```
yum install java-1.6.0-openjdk
```

After installing the JDK, run **/usr/sbin/alternatives** as **root** to insure that the proper JDK is available:

```
/usr/sbin/alternatives --config java
```

There are 3 programs which provide 'java'.

| Selection | Command |
|-----------|--------------------------------------------|
| ----- | |
| 1 | /usr/lib/jvm/jre-1.4.2-gcj/bin/java |
| + 2 | /usr/lib/jvm/jre-1.6.0-openjdk/bin/java |
| * 3 | /usr/lib/jvm/jre-1.6.0-sun.x86_64/bin/java |

2.2.2. Apache

Apache 2.x must be installed in order to install the TPS subsystem. Check that the appropriate version of Apache is installed.

```
yum info httpd
Installed Packages
Name      : httpd
Arch      : x86_64
Version   : 2.2.3
Release   : 22.el5_3.2
Size      : 2.9 M
Repo      : installed
...
```

Install Apache if it is not already available. For example:

```
yum install httpd
```

2.2.3. Red Hat Directory Server

All subsystems require access to Red Hat Directory Server 8.1 on the local machine or a remote machine. This Directory Server instance is used by the subsystems to store their system certificates and user data.

The Directory Server used by the Certificate System subsystems can be installed on Red Hat Enterprise Linux 5.3 32-bit, Red Hat Enterprise Linux 5.3 64-bit, or Solaris 9 Sparc 64-bit, regardless of the system on which Red Hat Certificate System is installed.

Check that the Red Hat Directory Server is already installed. For example:

```
yum info redhat-ds
Installed Packages
Name       : redhat-ds
Arch       : x86_64
Version    : 8.1.0
Release    : 0.14e15dsrv
Size       : 136M
Repo       : installed
...
```

Install and configure Red Hat Directory Server 8.1, if a directory service is not already available. For example:

```
yum install redhat-ds
setup-ds-admin.pl
```

Go through the configuration wizard; the default settings are fine for the Certificate System needs.

Installing Red Hat Directory Server is described in more detail in the *Red Hat Directory Server Installation Guide*.

2.2.4. Additional Packages

The following package groups and packages must be installed on all Red Hat Enterprise Linux systems:

- gnome-desktop (package group)
- compat-arch-support (package group)
- web-server (package group)
- kernel-smp (package)
- e2fsprogs (package)
- firefox (package)

To verify that the packages are installed, just run **rpm -qi** For example:

```
rpm -qi gnome-desktop
gnome-desktop-2.16.0-1.e15
```

On 64-bit Red Hat Enterprise Linux platforms, be certain that the 64-bit (x86_64) **compat-libstdc++** libraries are installed, and not only the 32-bit (i386) libraries. To confirm this, run the following as **root**:

```
rpm -qi compat-libstdc++ --queryformat '%{NAME}-%{VERSION}-%{RELEASE}.%{ARCH}.rpm\n' | grep x86_64
```

Numerous libraries should be displayed.

2.2.5. Firewall Configuration and iptables

Any firewalls must be configured to allow access to the Certificate System ports and to any other applications, like Red Hat Directory Server, which are required for the operation of the subsystems. Use caution when configuring the firewall, so that the system remains secure. The port numbers for the default instances are listed in [Section 9.5, “Default File and Directory Locations for Certificate System”](#).

As part of configuring the firewalls, if iptables is enabled, then it must have configured policies to allow communication over the appropriate Certificate System ports. Configuring iptables is described in the Red Hat Enterprise Linux *Deployment Guide*, such as [“Using iptables.”](#) Installing the subsystems will fail unless iptables is turned on and properly configured.

2.2.6. SELinux Settings

SELinux policies for Certificate System subsystems are installed as a dependency for Certificate System 8.0, in the **pki-selinux** package. The SELinux policies are automatically configured whenever a new instance is created by the **pkicreate** command.

Red Hat recommends running Certificate System with SELinux in **enforcing** mode, to make the most of the security policies.

If SELinux is set to **enforcing**, then any external modules or hardware which interact with the subsystems must be configured with the proper SELinux settings to proceed with subsystem installation:

- Third-party modules, such as for ECC or HSM, must have an SELinux policy configured for them, or SELinux needs to be changed from **enforcing** mode to **permissive** mode to allow the module to function. Otherwise, any subsystem operations which require the ECC module will fail.

Changing SELinux policies is covered in the *Red Hat Enterprise Linux Deployment Guide*, such as [chapter 46, “Customizing SELinux Policy.”](#)

- SELinux policies must be set for any nCipher netHSM 2000 modules, as described in [Section 2.5.2.4, “Setting up SELinux on nCipher netHSM 2000.”](#)

2.3. Packages Installed on Red Hat Enterprise Linux

Multiple packages are installed with the Certificate System, in addition to the core Certificate System components.

| RPMs for Certificate System Subsystems and Components | | |
|-------------------------------------------------------|---------------------------|-------------|
| osutil | pki-kra | pki-tks |
| pki-setup | pki-tps | |
| pki-ca | pki-migrate | |
| pki-common | pki-native-tools | symkey |
| pki-console | pki-ocsp | |
| pki-java-tools | | |
| RPMs for Tomcat Web Services | | |
| ant | jakarta-commons-discovery | jakarta-oro |

| | | |
|---------------------------------------------|----------------------------|---------------------------|
| avalon-framework | jakarta-commons-el | regex |
| avalon-logkit | jakarta-commons-fileupload | tomcat5 |
| axis | jakarta-commons-httpclient | tomcat5-common |
| bcel | jakarta-commons-launcher | tomcat5-jasper |
| classpathx-jaf | jakarta-commons-logging | tomcat5-server |
| classpathx-mail | jakarta-commons-modeler | velocity |
| eclipse-ecj | jakarta-commons-pool | werken.xpath |
| geronimo-specs | jdom | wsdl4j |
| xalan-j2 | | |
| geronimo-specs-compatible | jakarta-commons-beanutils | xerces-j2 |
| jakarta-commons-collections | ldapjdk | xml-commons |
| jakarta-commons-daemon | log4j | xml-commons-apis |
| jakarta-commons-dbc | mx4j | xml-commons-resolver |
| jakarta-commons-digester | | |
| RPMs for Apache Web Services | | |
| mod_perl | | perl-XML-Namespacesupport |
| pcre-devel | | perl-XML-Parser |
| perl-XML-SAX | | perl-Parser-RecDescent |
| tcl | | perl-XML-Simple |
| tcl-devel | | |
| RPMs for LDAP Support | | |
| cyrus-sasl | mozldap-devel | mozldap-tools |
| RPMs for SQL Lite Support for the RA | | |
| perl-DBD-SQLite | perl-DBI | sqlite-devel |
| RPMs for NSS and NSPR | | |
| jss | | |
| nspr | | |
| nss | | |
| svrcore | | |

2.4. Required Information for Subsystem Configuration

When the Certificate System subsystems are configured, some outside information must be available, as listed in [Table 2.2, “Required Information for Configuring Subsystems”](#).

Table 2.2. Required Information for Configuring Subsystems

| Information | Description |
|--------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Login PIN | There is a randomly-generated PIN in the preop.pin parameter in the CS.cfg file in the instance conf/ directory. This is used to log into the configuration wizard. |
| Security domain information | CAs can create a new security domain, which requires a unique name and a username and password for the CA agent who administers the domain. All other subsystems must join an existing security name. Have the username and password of the CA agent who administers the domain. |
| CA information | If the subsystem is not a CA, then it is necessary to select a CA from a drop-down menu or add an external CA. If a Certificate System CA is selected, then supply the CA agent username and password. |
| Subsystem information (for TPS configuration) | <p>When installing a TPS, you must have already configured the other subsystem and have their bind information available:</p> <ul style="list-style-type: none"> The CA The TKS (required) The DRM (optional, for server-side key generation) <p>When configuring the TPS, the TKS and DRM to connect with the TPS are selected from a drop-down list of all subsystems within the security domain.</p> |
| Directory Server hostname and port number | The Certificate System uses the user database of the Directory Server to store its information, and the hostname and port number of the LDAP directory is required for the Certificate System to access the database. |
| Directory Manager DN and password | The Certificate System must be able to bind to the user database, so a user ID and password must be supplied to bind to the Directory Server. This user is normally the Directory Manager. The default Directory Manager DN is cn=Directory Manager . |
| Certificate and key recovery files (for cloning) | If the subsystem being configured is a clone of another subsystem, then the backup files for the master subsystem must be locally accessible. |

2.5. Setting up Tokens for Storing Certificate System Subsystem Keys and Certificates

A subsystem instance generates and stores its key information in a key store, called a *token*. A subsystem instance can be configured for the keys to be generated and stored using the internal NSS token or on a separate cryptographic device, a hardware token.

2.5.1. Types of Hardware Tokens

A *token* is a hardware or software device that performs cryptographic functions and stores public-key certificates, cryptographic keys, and other data. The Certificate System defines two types of tokens, *internal* and *external*, for storing key pairs and certificates that belong to the Certificate System subsystems.

2.5.1.1. Internal Tokens

An internal (software) token is a pair of files, usually called the *certificate database* and *key database*, that the Certificate System uses to generate and store its key pairs and certificates. The Certificate System automatically generates these files in the filesystem of its host machine when first using the internal token. These files were created during the Certificate System subsystem configuration if the internal token was selected for key-pair generation.

In the Certificate System, the certificate database is named **cert8.db**; the key database is named **key3.db**. These files are located in the *instanceID/alias* directory.

2.5.1.2. External Tokens

An external token refers to an external hardware device, such as a smart card or hardware security module (HSM), that the Certificate System uses to generate and store its key pairs and certificates. The Certificate System supports any hardware tokens that are compliant with PKCS #11.

PKCS #11 is a standard set of APIs and shared libraries which isolate an application from the details of the cryptographic device. This enables the application to provide a unified interface for PKCS #11-compliant cryptographic devices.

The PKCS #11 module implemented in the Certificate System supports cryptographic devices supplied by many different manufacturers. This module allows the Certificate System to plug in shared libraries supplied by manufacturers of external encryption devices and use them for generating and storing keys and certificates for the Certificate System managers.

Consider using external tokens for generating and storing the key pairs and certificates used by Certificate System. These devices are another security measure to safeguard private keys because hardware tokens are sometimes considered more secure than software tokens.

Before using external tokens, plan how the external token is going to be used with the subsystem:

- All system keys for a subsystem must be generated on the same token.
- The subsystem keys must be installed in an empty HSM slot. If the HSM slot has previously been used to store other keys, then use the HSM vendor's utilities to delete the contents of the slot. The Certificate System has to be able to create certificates and keys on the slot with default nicknames. If not properly cleaned up, the names of these objects may collide with previous instances.
- A single HSM can be used to store certificates and keys for multiple subsystem instances, which may be installed on multiple hosts. When an HSM is used, any certificate nickname for a subsystem must be unique for every subsystem instance managed on the HSM.

2.5.1.3. Hardware Cryptographic Accelerators

The Certificate System can use hardware cryptographic accelerators with external tokens. Many of the accelerators provide the following security features:

- Fast SSL connections. Speed is important to accommodate a high number of simultaneous enrollment or service requests.
- Hardware protection of private keys. These devices behave like smart cards by not allowing private keys to be copied or removed from the hardware token. This is important as a precaution against key theft from an active attack of an online Certificate Manager.

2.5.2. Using Hardware Security Modules with Subsystems

The Certificate System supports the nCipher nethSM hardware security module (HSM) by default. Certificate System-supported HSMs are automatically added to the **secmod.db** database with **modutil** during the pre-configuration stage of the installation, if the PKCS #11 library modules are in the default installation paths.

During configuration, the **Key Store** panel displays the supported modules, along with the NSS internal software PKCS #11 module. All supported modules that are detected show a status of **Found** and is individually marked as either **Logged in** or **Not logged in**. If a token is found but not logged in, it is possible to log in using the **Login** under **Operations**. If the administrator can log into a token successfully, the password is stored in a configuration file. At the next start or restart of the Certificate System instance, the passwords in the password store are used to attempt a login for each corresponding token.

Administrators are allowed to select any of the tokens that are logged in as the default token, which is used to generate system keys.

2.5.2.1. Adding or Managing the HSM Entry for a Subsystem

When an HSM is selected as the default token, the following parameters are set in instance's **CS.cfg** file to configure the HSM:

```
#RHCS supported modules
preop.configModules.module0.commonName=NSS Internal PKCS #11 Module
preop.configModules.module0.imagePath=../img/mozilla.png
preop.configModules.module0.userFriendlyName=NSS Internal PKCS #11 Module
preop.configModules.module1.commonName=nfast
preop.configModules.module1.imagePath=../img/ncipher.png
preop.configModules.module1.userFriendlyName=nCipher's nFast Token Hardware Module
preop.configModules.module2.commonName=lunasa
preop.configModules.module2.imagePath=../img/safenet.png
preop.configModules.module2.userFriendlyName=SafeNet's LunaSA Token Hardware
Module
#selected token
preop.module.token=Internal Key Storage Token
```

In addition, the following parameter is set in the **password.conf** for the HSM password:

```
hardware-nethsm=caPassword
```

2.5.2.2. Using Chrysalis LunaSA HSM

To make sure that a LunaSA HSM works with Certificate System, edit the configuration files for the HSM *before* configuring the subsystems:

1. Check that the LunaSA module has been properly installed:

```
modutil -dbdir /var/lib/subsystem_name/alias -list
```

Listing of PKCS #11 Modules

```
-----
1. NSS Internal PKCS #11 Module
   slots: 2 slots attached
   status: loaded

       slot: NSS Internal Cryptographic Services
       token: NSS Generic Crypto Services

       slot: NSS User Private Key and Certificate Services
       token: NSS Certificate DB

2. lunasa
   library name: /usr/lunasa/lib/libCryptoki2_64.so
   slots: 1 slot attached
   status: loaded

       slot: LunaNet Slot
       token: lunasa3-ca
```

If the LunaSA module isn't listed, then install the module manually:

- a. Stop the subsystem.

```
service subsystem_name stop
```

- b. Load the module.

```
modutil -dbdir /var/lib/subsystem_name/alias -nocertdb -add lunasa -
libfile /usr/lunasa/lib/libCryptoki2_64.so
```

- c. Verify that the module has been loaded.

```
modutil -dbdir /var/lib/subsystem_name/alias -list
```

- d. Start the subsystem.

```
service subsystem_name start
```

2. Open the **/etc/Chrystoki.conf** configuration file.

3. Add this configuration parameter.

```
Misc { NetscapeCustomize=1023; }
```

4. If they are there, remove these two configuration lines for the applet version.

```
AppIdMajor=2;
AppIdMinor=4;
```

Then, after going through the subsystem configuration, but before restarting the server when completing the configuration wizard, edit the subsystem configuration to recognize the token:

1. Stop the server.

```
service subsystem_name stop
```

2. Edit the instance's **serverCertNick.conf** file in the **/var/lib/subsystem_name/conf** directory. Add the HSM token name to the **serverCert** parameter.

The original value only points to the server:

```
Server-Cert instanceID
```

The new value includes a reference to the LunaSA HSM:

```
lunasa3-ca:Server-Cert instanceID
```

3. Start the server.

```
service subsystem_name start
```

2.5.2.3. Installing External Tokens and Unsupported HSM

To use HSMs which are not officially supported by the Certificate System, add the module to the subsystem database manually. If the desired HSM does not appear in the **Security Modules** panel during the subsystem configuration, check that the HSM is installed and activated correctly. Then run **modutil** manually to add the module to the **secmod.db** database.

1. Install the cryptographic device, using the manufacturer's instructions. Be sure to name the token something that will help identify it easily later.
2. Install the PKCS #11 module using the **modutil** command-line utility.
 - a. Open the **alias** directory for the subsystem which is being configured with the PKCS #11 module. For example:

```
cd /var/lib/pki-ca/alias
```

- b. The required security module database file, **secmod.db**, should be created by default when the subsystem is created. If it does not exist, use the **modutil** utility to create **secmod.db**.

```
modutil -dbdir . -nocertdb -create
```

- c. Use the **modutil** utility to set the library information.

```
modutil -dbdir . -nocertdb / -add module_name -libfile library_file
```

library_file specifies the path to the library file containing the PKCS #11 interface module and **module_name** gives the name of the PKCS #11 module which was set when the drivers were installed.

- For the LunaSA HSM:

```
modutil -dbdir . -nocertdb -add lunasa -libfile  
/usr/lunasa/lib/libCryptoki2.so
```

- For an nCipher HSM:

```
modutil -dbdir . -nocertdb -add nethsm -libfile
/opt/nfast/toolkits/pkcs11/libcknfast.so
```

2.5.2.4. Setting up SELinux on nCiper netHSM 2000

SELinux policies are created and configured automatically for all Certificate System instances, so Certificate System can run with SELinux in enforcing or permissive modes.

If SELinux is in enforcing mode, than any hardware tokens to be used with the Certificate System instances must also be configured to run with SELinux in enforcing mode, or the HSM will not be available during subsystem installation.



IMPORTANT

SELinux must be configured for the HSM *before* installing any Certificate System instances.

1. Install the SELinux packages for Certificate System.

```
yum install pki-selinux
```

2. Reset the context of files in **/dev/nfast** to match the newly-installed policy.

```
/sbin/restorecon -R /dev/nfast
```

3. Restart the netHSM software.

2.5.3. Viewing Tokens

To view a list of the tokens currently installed for a Certificate System instance, use the **modutil** utility.

1. Open the instance **alias** directory. For example:

```
cd /var/lib/pki-ca/alias
```

2. Show the information about the installed PKCS #11 modules installed as well as information on the corresponding tokens using the **modutil** tool.

```
modutil -dbdir . -nocertdb -list
```

2.5.4. Detecting Tokens

To see if a token can be detected by Certificate System, use the **TokenInfo** utility. This is a Certificate System tool which is available after the Certificate System packages are installed.

```
TokenInfo
```

This utility will return all tokens which can be detected by the Certificate System, not only tokens which are installed in the Certificate System.

Chapter 3. Installation and Configuration

The Certificate System is comprised of subsystems which can be independently installed on different servers, multiple instances installed on a single server, and other flexible configurations for availability, scalability, and failover support. The procedures for downloading, installing, and configuring instances of Certificate System subsystems are described in this chapter.

The Certificate System servers include six subsystems:

- Certificate Authority (CA)
- Registration Authority (RA)
- Data Recovery Manager (DRM), sometimes referred to as a Key Recovery Authority (KRA)
- Online Certificate Status Protocol (OCSP) Responder
- Token Key Service (TKS)
- Token Processing System (TPS)

The Certificate System client is the Enterprise Security Client. For information about the Enterprise Security Client, see the *Certificate System Enterprise Security Client Guide*.

3.1. Overview of Installation

The individual subsystems for Red Hat Certificate System are installed and then configured individually. The initial installation is done using package management tools such as RPM; the subsystem setup is done through an HTML-based configuration wizard.

1. Install a Red Hat Directory Server. This can be on a different machine from the Certificate System, which is the recommended scenario for most deployments.
2. Download the Certificate System packages from the Red Hat Network channel. Each subsystem has its own packages, as well as dependencies and related packages. These are listed in [Section 2.3, “Packages Installed on Red Hat Enterprise Linux”](#).
3. Install the packages, as described in [Section 3.2, “Installing the Certificate System Packages”](#).
By default, the installation process immediately launches **pkicreate** to create the default instances as soon as the subsystem packages are installed. The default instances are configured with the default settings listed in [Section 9.5, “Default File and Directory Locations for Certificate System”](#). It is also possible to prevent the **pkicreate** command from running so that you can configure an instance with custom settings, which is described in [Chapter 5, Creating Additional Subsystem Instances](#).
4. Configure the Certificate System CA subsystem. At least one CA subsystem *must* be installed and fully configured before any other type of subsystem can be configured.
See [Section 3.3, “Configuring a CA”](#) for instructions on setting up the Certificate Manager.
5. Configure the RA, OCSP, DRM, and TKS subsystems. Once the CA is installed, the other subsystems, except for the TPS, can be installed and configured in any order.
See [Section 3.5, “Configuring a DRM, OCSP, or TKS”](#) and [Section 3.4, “Configuring an RA”](#) for the process on installing and configuring the OCSP, DRM, TKS, and RA subsystems.
6. Configure the TPS subsystem. The TPS requires having an existing TKS and DRM available when it is configured, so this is the last subsystem to set up.
See [Section 3.6, “Configuring a TPS”](#) for the process on installing and configuring the TPS.

The order in which subsystems are configured is very important because of the basic relationships which are established between subsystems at the time they are installed. For example, every subsystem depends on a certificate authority; the TPS also depends on a TKS and (optionally) DRM.

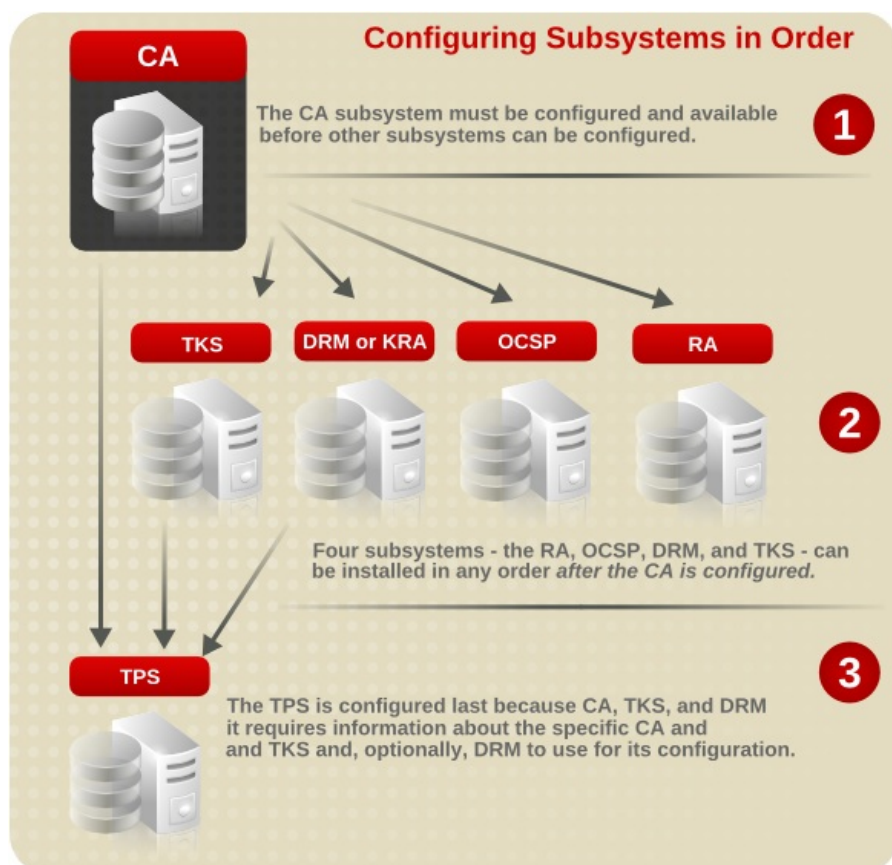


Figure 3.1. Order of Subsystem Configuration

3.2. Installing the Certificate System Packages

There are two ways to obtain and install the subsystem packages. For all supported platforms, the Certificate System packages can be downloaded as ISO images through the appropriate Red Hat Network channel. These packages are then installed through a package utility, such as **rpm**.

Alternatively, if the appropriate network access is available, the subsystems and dependencies can be downloaded and installed on Red Hat Enterprise Linux systems using the **yum** command.

Several packages are installed with the Certificate System packages for related applications and dependencies, not only for the subsystems. These packages are listed in [Section 2.3, “Packages Installed on Red Hat Enterprise Linux”](#).

- [Section 3.2.1, “Installing through yum”](#)
- [Section 3.2.2, “Installing from an ISO Image”](#)

NOTE

When the first subsystem is installed on a machine, the installation process automatically creates a new user (**pkiuser**) and group (**pkiuser**). All default Certificate System instances run as this user and group.

3.2.1. Installing through yum

 **NOTE**

pkicreate is launched by the installer to create the default instances, using default settings. There is an environment variable, **DONT_RUN_PKICREATE**, which stops the **pkicreate** script from running automatically after the subsystems are installed. Setting **DONT_RUN_PKICREATE** allows the default instances to be installed in user-defined installation directories, instead of the default locations in **/var/lib**. It can be preferable to install through the ISO image with this environment variable set to block the **pkicreate** script for deployments where the default instances must be installed in custom locations.

 **NOTE**

To use an IPv6 hostname for configuration, set the hostname in the **PKI_HOSTNAME** environment variable before installing the packages. This is described in [Section 4.4, “Enabling IPv6 for a Subsystem”](#).

To install the initial subsystems on Red Hat Enterprise Linux 5 (32-bit), run a command like the following for each subsystem:

```
yum install pki-subsystem
```

 **NOTE**

yum is used only for the first subsystem instance; any additional subsystem instances are added using **pkicreate**.

subsystem can be any of the Certificate System subsystems:

- ▶ **ca** for the Certificate Manager.
- ▶ **ra** for the Registration Authority.
- ▶ **kra** for the Data Recovery Manager.
- ▶ **ocsp** for the Online Certificate Status Protocol Responder.
- ▶ **tk**s for the Token Key System.
- ▶ **tps** for the Token Processing System.
- ▶ **console** for the Java console.

Once the packages are installed, then the installer automatically launches the **pkicreate** script to create the default subsystem instance automatically. A URL to access the new instance is printed to the screen which gives the subsystem instances hostname, port, and a login PIN to access the configuration wizard.

```
PKI instance creation Utility ...
```

```
PKI instance creation completed ...
```

```
Starting instance_name: [ OK ]
```

```
instance_name (pid 17990) is running ...
```

```
'instance_name' must still be CONFIGURED!
(see /var/log/instance_name-install.log)
```

Before proceeding with the configuration, make sure the firewall settings of this machine permit proper access to this subsystem.

Please start the configuration by accessing:

```
https://hostname.domainname:admin-port/subsystem_type/admin/console/config/login?
pin=pin
```

After configuration, the server can be operated by the command:

```
/sbin/service instance_name start | stop | restart
```

To install the **pki-console** to administer the subsystems, run the following:

```
yum install pki-console
```

3.2.2. Installing from an ISO Image

Red Hat Certificate System 8.0 can also be downloaded from Red Hat Network as an ISO image. This ISO image contains an **RPMS/** directory which can be used as a local yum repository.

1. Open the **Red Hat Certificate System 8.0** Red Hat Network channel and download the ISO image.
2. Place that **RPMS/** directory on a web server and then configure yum to use that location as a repository.
3. Install Certificate System as described in [Section 3.2.1, “Installing through yum”](#), including setting any environment variables, HSMs, or IPv6 settings.

3.3. Configuring a CA

The CA is always the first subsystem to be configured; every other subsystem depends on the CA for its configuration. The CA, along with setting up the CA hierarchy for the PKI, issues certificates which every subsystem uses to function and sets up a security domain which establishes trusted relationships between subsystems.



NOTE

If a CA has an ECC signing certificate, it can issue both RSA and ECC client certificates. To enable ECC for the CA, load an ECC module first (described in [Section 4.2, “Installing a CA with ECC Enabled”](#)) and then configure the CA.

Subsystem configuration is done by accessing a unique web-based configuration page for the instance. The only supported web browser for subsystem configuration is Mozilla Firefox.

1. Install the subsystem packages. For example:

```
yum install pki-ca
```

Once the packages are installed, then the installer automatically launches the **pkicreate** script to create the default subsystem instance automatically. A URL to access the new instance is printed to the screen which gives the subsystem instances hostname, port, and a login PIN to access the configuration wizard.

```
http://server.example.com:9180/ca/admin/console/config/login?pin=
```

2. Open the configuration wizard using the URL returned from the package installation.
Alternatively, log into the setup wizard through admin link on the services page and supply the **preop.pin** value from the **/var/lib/pki-ca/conf/CS.cfg** file when prompted.

```
https://server.example.com:9444/ca/services
```

3. Create a new security domain.

Security Domain

securitydomain

A security domain is a registry for all of the PKI services within an enterprise. Applications may use the security domain to locate other PKI services. [\[Details\]](#)

☒ **Create a New Security Domain**
If no security domain exists, a new one must be created for this CA.

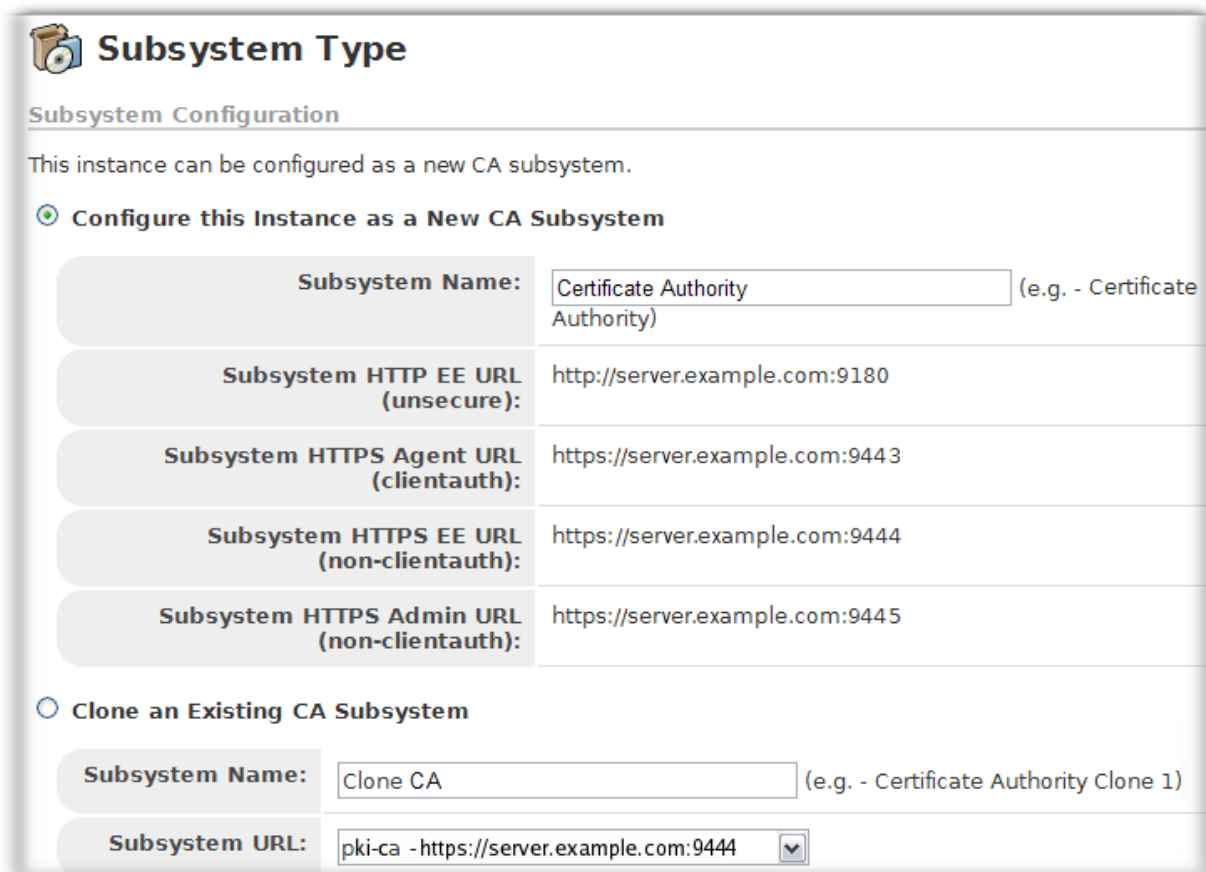
| | |
|----------------------------------------------------|-------------------------------------------------|
| Security Domain Name: | Example Domain (e.g. - Red Hat Security Domain) |
| Subsystem HTTP EE URL (unsecure): | http://server.example.com:9180 |
| Subsystem HTTPS Agent URL (clientauth): | https://server.example.com:9443 |
| Subsystem HTTPS EE URL (non-clientauth): | https://server.example.com:9444 |
| Subsystem HTTPS Admin URL (non-clientauth): | https://server.example.com:9445 |

☐ **Join an Existing Security Domain**
Enter the URL to an existing security domain.

| | |
|----------------------------------------------------------|-------------------------------------------------------------------|
| Security Domain HTTPS Admin URL (non-clientauth): | https://server.example.com:9445 (e.g. - https://example.com:9445) |
|----------------------------------------------------------|-------------------------------------------------------------------|

The default CA instance must create a new security domain. Subsequent CAs can create a new domain or join an existing security domain, but it is recommended that each CA have its own security domain.

4. Enter a name for the new instance.



Subsystem Type

Subsystem Configuration

This instance can be configured as a new CA subsystem.

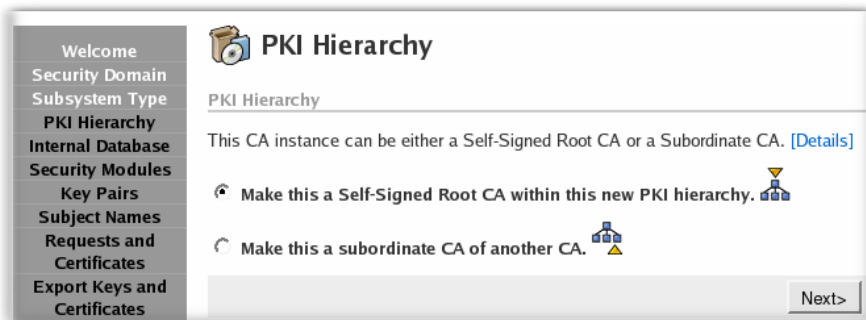
☒ **Configure this Instance as a New CA Subsystem**

| | |
|----------------------------------------------------|-----------------------------------------------------------------------------------|
| Subsystem Name: | <input type="text" value="Certificate Authority"/> (e.g. - Certificate Authority) |
| Subsystem HTTP EE URL (unsecure): | <input type="text" value="http://server.example.com:9180"/> |
| Subsystem HTTPS Agent URL (clientauth): | <input type="text" value="https://server.example.com:9443"/> |
| Subsystem HTTPS EE URL (non-clientauth): | <input type="text" value="https://server.example.com:9444"/> |
| Subsystem HTTPS Admin URL (non-clientauth): | <input type="text" value="https://server.example.com:9445"/> |

☐ **Clone an Existing CA Subsystem**

| | |
|------------------------|------------------------------------------------------------------------------|
| Subsystem Name: | <input type="text" value="Clone CA"/> (e.g. - Certificate Authority Clone 1) |
| Subsystem URL: | <input type="text" value="pki-ca - https://server.example.com:9444"/> ▼ |

- Set up the PKI hierarchy. Commonly, the first CA is a root, or self-signed, CA, meaning that it signs its own CA signing certificate rather than submitting its certificates to a third-party CA for issuance. Subsequent CAs can be subordinate CAs to that root. There are many other options, depending on the PKI environment.



Welcome
Security Domain
Subsystem Type
PKI Hierarchy
Internal Database
Security Modules
Key Pairs
Subject Names
Requests and Certificates
Export Keys and Certificates

PKI Hierarchy

PKI Hierarchy

This CA instance can be either a Self-Signed Root CA or a Subordinate CA. [\[Details\]](#)

☒ **Make this a Self-Signed Root CA within this new PKI hierarchy.**


☐ **Make this a subordinate CA of another CA.**

For a CA, there are two possible configuration options:

- **Root CA.** A root CA signs its own CA signing certificate and, therefore, can set its own certificate issuance rules.
- **Subordinate CA.** A subordinate CA receives its CA signing certificate from a root CA. The root CA must be referenced here; it can be another Certificate System CA, but, for the default (i.e., first) CA instance, this will probably be an external root CA. The certificate requests generated in this process must be submitted to the external CA and be approved before configuration can be completed.

See the planning question [Should the Certificate Manager be a self-signed root CA or a subordinate CA?](#)

- Fill in the information for the LDAP server which will be used for the instance's internal database. This requires connection information for the Directory Server instance, such as the hostname, port number, bind DN (username), and password. This step also creates a database in the Directory Server and a corresponding base directory entry (base DN) to use for the subsystem's entries.



Internal Database

Please provide information to an existing Red Hat Directory Server that can be used as the internal database for this instance. [\[Details\]](#)

Note: If the Red Hat Directory Server is at a remote host, it is highly recommended that SSL should be used.

| | |
|----------------|---------------------------------------------------------------|
| Host: | <input type="text" value="localhost"/> |
| Port: | <input type="text" value="389"/> <input type="checkbox"/> SSL |
| Base DN: | <input type="text" value="dc=server.example.com-pki-ca"/> |
| Database: | <input type="text" value="server.example.com-pki-ca"/> |
| Bind DN: | <input type="text" value="cn=Directory Manager"/> |
| Bind Password: | <input type="password" value="....."/> |

☐ Remove the existing data from the **Base DN** shown above.

The hostname can be the fully-qualified domain name or an IPv4 or IPv6 address, if IPv6 was configured before the packages were installed.



NOTE

One thing that can derail subsystem configuration or function is having services that are unable to connect with each other. If servers that need to communicate with each other are on different servers or networks, when the firewalls and iptables must be configured to give the required access.

If the Red Hat Directory Server instances is on a different server or network than the Certificate System subsystem, then make sure that the Certificate System host's firewall allows access to whatever LDAP port was set in the previous configuration panel. Installation will not complete if iptables is not configured properly. To configure iptables, see the Red Hat Enterprise Linux *Deployment Guide*, such as ["Using iptables."](#) It is also possible to simply turn iptables off.

7. Select the token which will store the Certificate System certificates and keys; a list of detected hardware tokens and databases is given.



IMPORTANT

Any hardware tokens used with the instance must be configured *before* configuring the subsystem instance. If the HSM is not properly configured, it may not be listed in the key stores panel or the instance may not function properly. HSM configuration is described in [Section 2.5.2, "Using Hardware Security Modules with Subsystems"](#).

To determine whether a token is detected by the Certificate System, use the **TokenInfo** tool, as described in [Section 2.5.4, "Detecting Tokens"](#).



Key Store

Two lists of security modules are provided below. The **Supported Security Modules** list consists of both software-based and hardware-based security modules that this PKI solution supports, while the **Other Security Modules** list consists of any other security modules found by this PKI subsystem that are not recognized as one of the supported security modules. [\[Details\]](#)

Supported Security Modules

| Module/Token | Status | Default | Operations |
|----------------------------------------|---------------|---------|-----------------------|
| NSS Internal PKCS #11 Module | Found | | |
| - Internal Key Storage Token | Logged In | | |
| nCipher's nFast Token Hardware Module | Found | | |
| - accelerator | Not logged In | | Login |
| - nethsm | Not logged In | | Login |
| SafeNet's LunaSA Token Hardware Module | Found | | |
| - lunasa1-ca | Not logged In | | Login |
| - lunasa2-ca | Not logged In | | Login |

Other Security Modules

The security modules listed below are modules found by the server but not recognized as one of the supported modules. If the user believes that any listed modules below should have been supported, please check the "CS.cfg" configuration file to see if there is a name mismatch and adjust this accordingly.

The Certificate System automatically discovers Safenet's LunaSA and nCipher's netHSM hardware security modules. The discovery process assumes that the client software installations for these modules are local to the Certificate System subsystem and are in the following locations:

- » LunaSA: `/usr/lunasa/lib/libCryptoki2.so`
- » nCipher: `/opt/nfast/toolkits/pkcs11/libcknfast.so`

8. Set the key size and the hashing algorithm to use. By default, the settings for the signing key are applied to the keys for every certificate for the CA. To set different key types, sizes, or hashing algorithms for each certificate, click the **[Advanced]** link to expand the form so each key pair is listed.

The default RSA key size is 2048 and for ECC, 256.



Key Pairs

Select the key pair type(s) and associated key pair size(s) and hashing algorithm(s) from the pulldown menus. [\[Details\]](#)

[\[Advanced\]](#)

Common Key Settings

Key Type:

RSA

Hashing
Algorithm:

SHA256withRSA

☒ Use the default key size (2048 bits for RSA, 256 bits for ECC).

☐ Use the following custom key size:

Key Size:

2048



NOTE

An ECC CA signing certificate can be used to sign both ECC and RSA certificates. If you do not want to use the ECC client certificate that is generated at installation, simply replace the client certificate after configuration, and keep the ECC CA signing certificate.

An ECC module must be loaded for ECC certificates to be generated. Adding ECC support is covered in [Section 4.2, “Installing a CA with ECC Enabled”](#). Any ECC-enabled PKCS#11 module must be loaded *before* beginning to configure the CA.

The hashing algorithms that are available depend on whether RSA or ECC is selected as the key type. For RSA, the available algorithms are as follows:

- SHA256withRSA (the default)
- SHA1withRSA
- SHA256withRSA
- SHA512withRSA
- MD5withRSA
- MD2withRSA

For ECC:

- SHA256withEC (the default)
- SHA1withEC
- SHA384withEC
- SHA512withEC

9. Optionally, change the subject names for the certificates.



Subject Names

Each certificate associated with this instance needs to have a unique name within the PKI hierarchy. The following information will be used to generate these unique names. [\[Details\]](#)

CA Signing Certificate

DN:

Nickname:

OCSP Signing Certificate

DN:




NOTE

Certificate nicknames must be unique, and changing the default nicknames is one way to ensure that.

Having unique certificate nicknames is vital for using an HSM, since any nickname conflicts (even for subsystems on different servers) will cause configuration to fail.

10. The next panels generate and show certificate requests, certificates, and key pairs.



Requests and Certificates

A certificate signing request (CSR) contains a public key and is an unsigned copy of the certificate.


If a given CSR has been successfully signed by a CA, then the certificate will be designated below by a certificate icon labeled Certificate Generated Successfully.

However, if a given CSR contains an **action required** label under its certificate icon, then those requests must be *manually* submitted to a CA for certificate generation.

Press the [Apply] button after certificates and chains are pasted in.

Press the [Next] button once all certificates have been generated successfully.

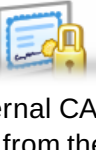
CN=Certificate Authority,O= Example Domain



Certificate Generated Successfully

[View Certificate Request \(CSR\)](#)
[View Certificate in Base64-Encoding](#)
[View Certificate Pretty Print](#)

CN=OCSP Signing Certificate,O= Example Domain



Certificate Generated Successfully

[View Certificate Request \(CSR\)](#)
[View Certificate in Base64-Encoding](#)

If an external CA is used to issue the certificates, configuration cannot go forward until they are received from the external CA. When they are issued, paste the certificates into this panel to add them to the CA database, and then proceed with the installation. Click **Apply** to view the certificates as they are imported.

11. If the subsystem will ever be cloned, or as a protection if keys or certificates are ever lost, back up the keys and certificates when prompted. It is also possible to extract these keys later, as long as they are not stored on an HSM.

**NOTE**

It is not possible to export keys and certificates stored on an HSM to a **.p12** file. It is also not necessary to extract keys from the HSM to clone a subsystem. The keys are already stored on the HSM and accessible to any cloned instances.

**Export Keys and Certificates**Export Keys and Certificates

To setup a cloned subsystem, the master subsystem's keys and certificates (with the exception of the SSL server key and certificate) as well as the CA certificate chains need to be exported, and later imported into the cloned subsystem. All of these keys and certificates are stored in a single file in the PKCS #12 format which is protected by the password specified below. This export operation is performed only when the master subsystem's keys and certificates are stored in the software token.

If these keys and certificates are stored in a hardware token, the hardware token vendor needs to be consulted for information on how to export them.

☒ **Export subsystem keys and certificates**

Password to protect the PKCS #12 file:

Password again:

☐ **Don't export subsystem keys and certificates****Next>**

12. Provide the information for the new subsystem administrator.

**Administrator**

The administrator is the privileged user who manages this subsystem. Please enter the following relevant information. A certificate request will be automatically generated and submitted. The administrator entry will be created in the internal database and his certificate will be imported into this browser automatically in the next panel. This administrator certificate is used to access the agent interface of this subsystem.

UID:

Name:

Email:

Password:

Password (Again):

13. Click **Next** through the remaining panels to import the agent certificate into the browser and complete the configuration.
14. When the configuration is complete, restart the subsystem.

```
service pki-ca restart
```

**IMPORTANT**

The new instance is not active until it is restarted, and weird behaviors can occur if you try to use the instance without restarting it first.

3.4. Configuring an RA

Subsystem configuration is done by accessing a unique web-based configuration page for the instance. The only supported web browser for subsystem configuration is Mozilla Firefox.

**IMPORTANT**

Before any RA can be set up, a Certificate System CA must be installed, configured, and running. This subsystem depends on the CA to issue their certificates and to create a security domain. If the security domain CA is not available, then the configuration process fails.

1. Install the subsystem packages. For example:

```
yum install pki-ra
```

Once the packages are installed, then the installer automatically launches the **pkicreate** script to create the default subsystem instance automatically. A URL to access the new instance is printed to the screen which gives the subsystem instances hostname, port, and a login PIN to access the configuration wizard.

```
https://server.example.com:12889//ra/admin/console/config/login?
pin=ki7E1MByNIUcPJ6RKHmH
```

2. Open the configuration wizard using the URL returned by the package installation.

```
https://server.example.com:12889//ra/admin/console/config/login?
pin=ki7E1MByNIUcPJ6RKHmH
```

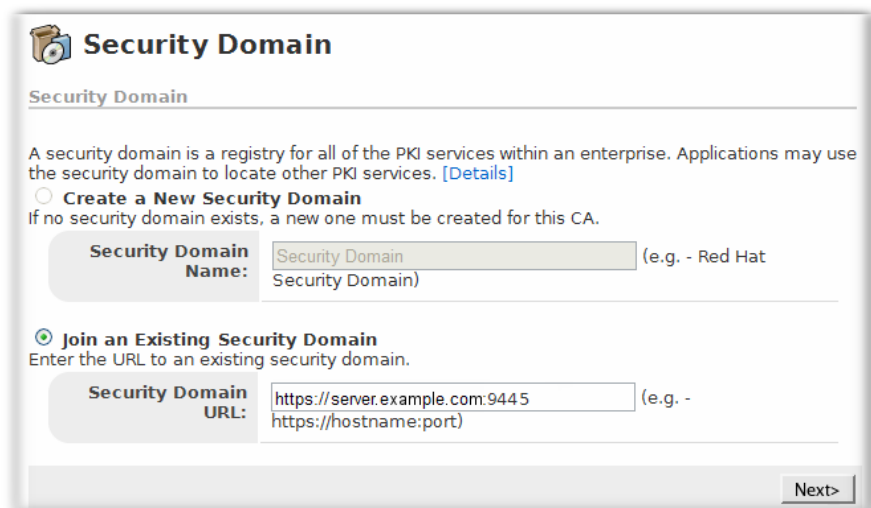
Alternatively, log into the setup wizard through admin link on the services page and supply the **preop.pin** value from the **/var/lib/pki-ra/conf/CS.cfg** file when prompted.

```
https://server.example.com:12889/services
```

3. Join an existing security domain by entering the CA information. This URL can be identified by running **service pki-ca status** on the CA's host; the security domain URL is returned with the other configuration settings. For example:

```
https://server.example.com:9445
```

When the CA is successfully contacted, then supply the admin username and password for the CA so that it can be properly accessed.



Security Domain

Security Domain

A security domain is a registry for all of the PKI services within an enterprise. Applications may use the security domain to locate other PKI services. [\[Details\]](#)

☐ **Create a New Security Domain**
If no security domain exists, a new one must be created for this CA.

Security Domain Name: (e.g. - Red Hat Security Domain)

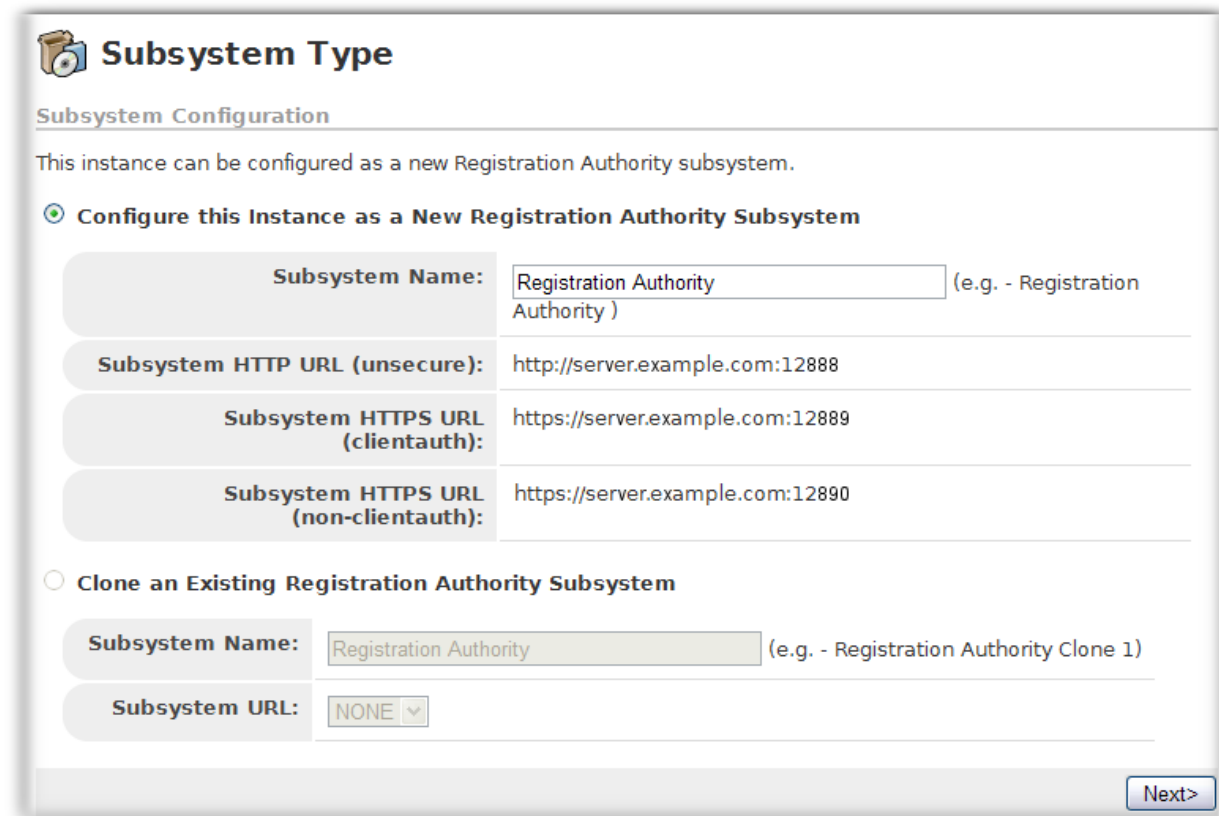
☒ **Join an Existing Security Domain**
Enter the URL to an existing security domain.

Security Domain URL: (e.g. - https://hostname:port)

Next>

The hostname for the security domain CA can be the fully-qualified domain name or an IPv4 or IPv6 address, if IPv6 was configured before the packages were installed.

4. Enter a name for the new instance.



Subsystem Type

Subsystem Configuration

This instance can be configured as a new Registration Authority subsystem.

☒ **Configure this Instance as a New Registration Authority Subsystem**

| | |
|----------------------------------------------|--------------------------------------------------------------------------------------|
| Subsystem Name: | <input type="text" value="Registration Authority"/> (e.g. - Registration Authority) |
| Subsystem HTTP URL (unsecure): | <input type="text" value="http://server.example.com:12888"/> |
| Subsystem HTTPS URL (clientauth): | <input type="text" value="https://server.example.com:12889"/> |
| Subsystem HTTPS URL (non-clientauth): | <input type="text" value="https://server.example.com:12890"/> |

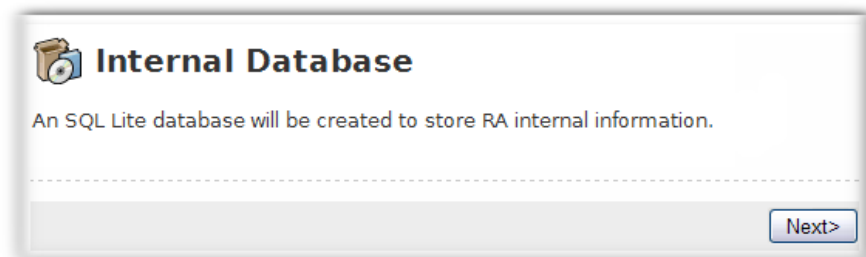
☐ **Clone an Existing Registration Authority Subsystem**

| | |
|------------------------|---------------------------------------------------------------------------------------------|
| Subsystem Name: | <input type="text" value="Registration Authority"/> (e.g. - Registration Authority Clone 1) |
| Subsystem URL: | <input type="text" value="NONE"/> |

Next>

5. Select the CA which will issue, renew, and revoke certificates for certificates processed through the RA. All of the CAs configured in the security domain are listed in a dropdown menu.

6. Click **Next** on the **Internal Database** panel; the SQLite database is created automatically.



NOTE

The RA uses a SQLite database to store its configuration and user data rather than an LDAP database, as the other subsystems do.

7. Select the token which will store the Certificate System certificates and keys; a list of detected hardware tokens and databases is given.



IMPORTANT

Any hardware tokens used with the instance must be configured *before* configuring the subsystem instance. If the HSM is not properly configured, it may not be listed in the key stores panel or the instance may not function properly. HSM configuration is described in [Section 2.5.2, “Using Hardware Security Modules with Subsystems”](#).

To determine whether a token is detected by the Certificate System, use the **TokenInfo** tool, as described in [Section 2.5.4, “Detecting Tokens”](#).

Key Store

Two lists of security modules are provided below. The **Supported Security Modules** list consists of both software-based and hardware-based security modules that this PKI solution supports, while the **Other Security Modules** list consists of any other security modules found by this PKI subsystem that are not recognized as one of the supported security modules. [\[Details\]](#)

Supported Security Modules

| Module/Token | Status | Default | Operations |
|----------------------------------------|---------------|---------|-----------------------|
| NSS Internal PKCS #11 Module | Found | | |
| - Internal Key Storage Token | Logged In | | |
| nCipher's nFast Token Hardware Module | Found | | |
| - accelerator | Not logged In | | Login |
| - nethsm | Not logged In | | Login |
| SafeNet's LunaSA Token Hardware Module | Found | | |
| - lunasa1-ca | Not logged In | | Login |
| - lunasa2-ca | Not logged In | | Login |

Other Security Modules

The security modules listed below are modules found by the server but not recognized as one of the supported modules. If the user believes that any listed modules below should have been supported, please check the "CS.cfg" configuration file to see if there is a name mismatch and adjust this accordingly.

The Certificate System automatically discovers Safenet's LunaSA and nCipher's netHSM hardware security modules. The discovery process assumes that the client software installations

for these modules are local to the Certificate System subsystem and are in the following locations:

- LunaSA: `/usr/lunasa/lib/libCryptoki2.so`
- nCipher: `/opt/nfast/toolkits/pkcs11/libcknfast.so`

8. Set the key size. The default RSA key size is 2048.



Key Pairs

Select the key pair type(s) and associated key pair size(s) from the pulldown menus. [\[Details\]](#)

[\[Advanced\]](#)

Common Key Settings

Key Type: RSA ▼

☒ Use the default key size (2048 bits for RSA, 256 bits for ECC).

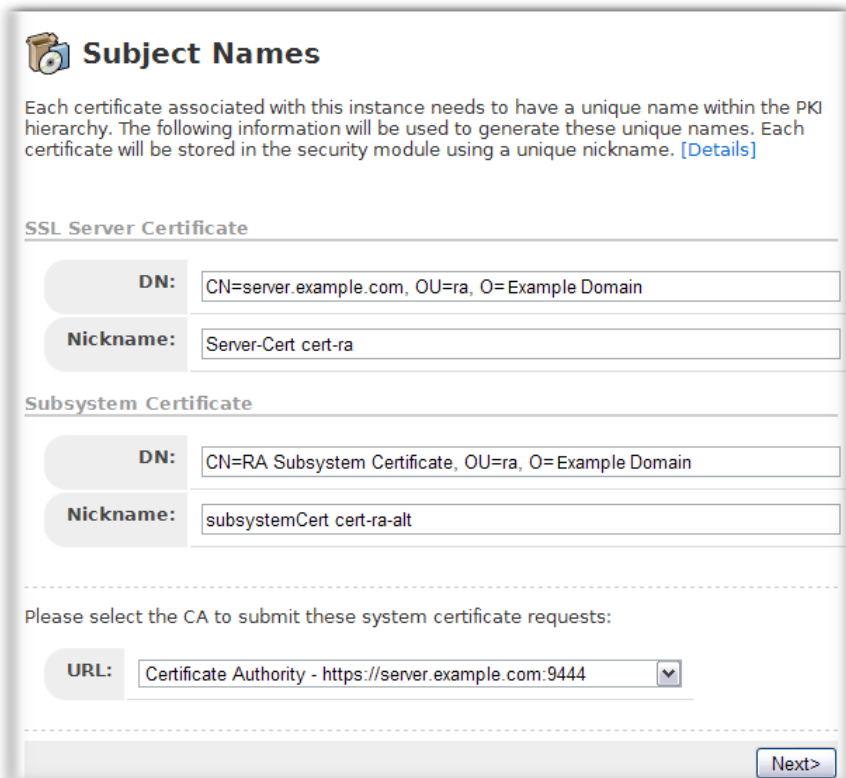
☐ Use the following custom key size:

Key Size: 2048

Note: After pressing Next, keys will be generated on the server, which will take some time to complete. Please wait for the next panel to appear.

Next>

9. Optionally, change the subject names for the certificates.



Subject Names

Each certificate associated with this instance needs to have a unique name within the PKI hierarchy. The following information will be used to generate these unique names. Each certificate will be stored in the security module using a unique nickname. [\[Details\]](#)

SSL Server Certificate

DN: CN=server.example.com, OU=ra, O=Example Domain

Nickname: Server-Cert cert-ra

Subsystem Certificate

DN: CN=RA Subsystem Certificate, OU=ra, O=Example Domain

Nickname: subsystemCert cert-ra-alt

Please select the CA to submit these system certificate requests:

URL: Certificate Authority - https://server.example.com:9444 ▼

Next>




NOTE

Certificate nicknames must be unique, and changing the default nicknames is one way to ensure that.

Having unique certificate nicknames is vital for using an HSM, since any nickname conflicts (even for subsystems on different servers) will cause configuration to fail.

10. The next panels generate and show certificate requests, certificates, and key pairs.



Requests and Certificates


A certificate signing request (CSR) contains a public key and is an unsigned copy of the certificate.

If a given CSR has been successfully signed by a CA, then the certificate will be designated below by a certificate icon labeled Certificate Generated Successfully.

However, if a given CSR contains an **action required** label under its certificate icon, then those requests must be *manually* submitted to a CA for certificate generation. [\[Details\]](#)

Press the [Next] button once all certificates have been generated successfully.

SSL Server Certificate




Certificate Generated Successfully

[View Certificate Request \(CSR\)](#)

[View Certificate in Base64-Encoding](#)

[View Certificate Pretty Print](#)

Subsystem Certificate



Certificate Generated Successfully


[View Certificate Request \(CSR\)](#)

[View Certificate in Base64-Encoding](#)

[View Certificate Pretty Print](#)

If an external CA is used to issue the certificates, configuration cannot go forward until they are received from the CA. When they are issued, paste the certificates into this panel to add them to the subsystem database, and then proceed with the installation. Click **Apply** to view the certificates as they are imported.

11. Provide the information for the new subsystem administrator.



Administrator

The administrator is the privileged user who manages this subsystem. Please enter the following relevant information. A certificate request will be automatically generated and submitted. The administrator entry will be created in the internal database and his certificate will be imported into this browser automatically in the next panel. This administrator certificate is used to access the agent interface of this subsystem.

| | |
|-------------------|------------------------------------------------------------------|
| UID: | <input type="text" value="admin"/> |
| Name: | <input type="text" value="RA Administrator of Instance pki-ra"/> |
| Email: | <input type="text" value="email@example.com"/> |
| Password: | <input type="password" value="*****"/> |
| Password (Again): | <input type="password" value="*****"/> |

12. Click **Next** through the remaining panels to import the agent certificate into the browser and complete the configuration.
13. When the configuration is complete, restart the subsystem.

```
service pki-ra restart
```

**IMPORTANT**

The new instance is not active until it is restarted, and weird behaviors can occur if you try to use the instance without restarting it first.

3.5. Configuring a DRM, OCSP, or TKS

Subsystem configuration is done by accessing a unique web-based configuration page for the instance. The only supported web browser for subsystem configuration is Mozilla Firefox.

**IMPORTANT**

Before any DRM, OCSP, or TKS subsystem can be set up, a Certificate System CA must be installed, configured, and running. These subsystems depend on the CA to issue their certificates and to create a security domain. If the security domain CA is not available, then the configuration process fails.

1. Install the subsystem packages. For example, to install a DRM:

```
yum install pki-kra
```

**NOTE**

A *Data Recovery Manager* (DRM) is also known as a *Key Recovery Agent* (KRA). All command-line tools and many files for the DRM use the abbreviation **kra** for this reason. In the documentation, the subsystem used to archive and recover keys is called the DRM or KRA interchangeably.

Once the packages are installed, then the installer automatically launches the **pkicreate** script to create the default subsystem instance automatically. A URL to access the new instance is printed to the screen which gives the subsystem instances hostname, port, and a login PIN to access the configuration wizard.

```
http://server.example.com:10180/kra/admin/console/config/login?
pin=ki7E1MByNIUcPJ6RKHmH
```

2. Open the configuration wizard using the URL returned by the package installation.

```
http://server.example.com:10180/kra/admin/console/config/login?
pin=ki7E1MByNIUcPJ6RKHmH
```

Alternatively, log into the setup wizard through admin link on the services page and supply the **preop.pin** value from the `/var/lib/subsystem_name/conf/CS.cfg` file when prompted.

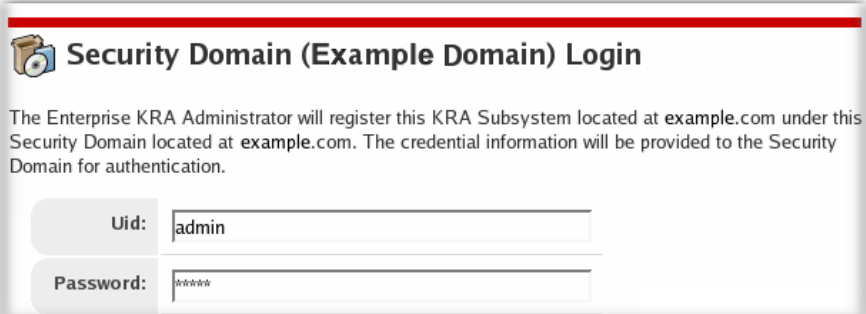
```
https://server.example.com:10444/kra/services
```

3. Join an existing security domain by entering the CA information. This URL can be identified by running **service pki-ca status** on the CA's host; the security domain URL is returned with

the other configuration settings. For example:

`https://server.example.com:9445`

When the CA is successfully contacted, then supply the admin username and password for the CA so that it can be properly accessed.



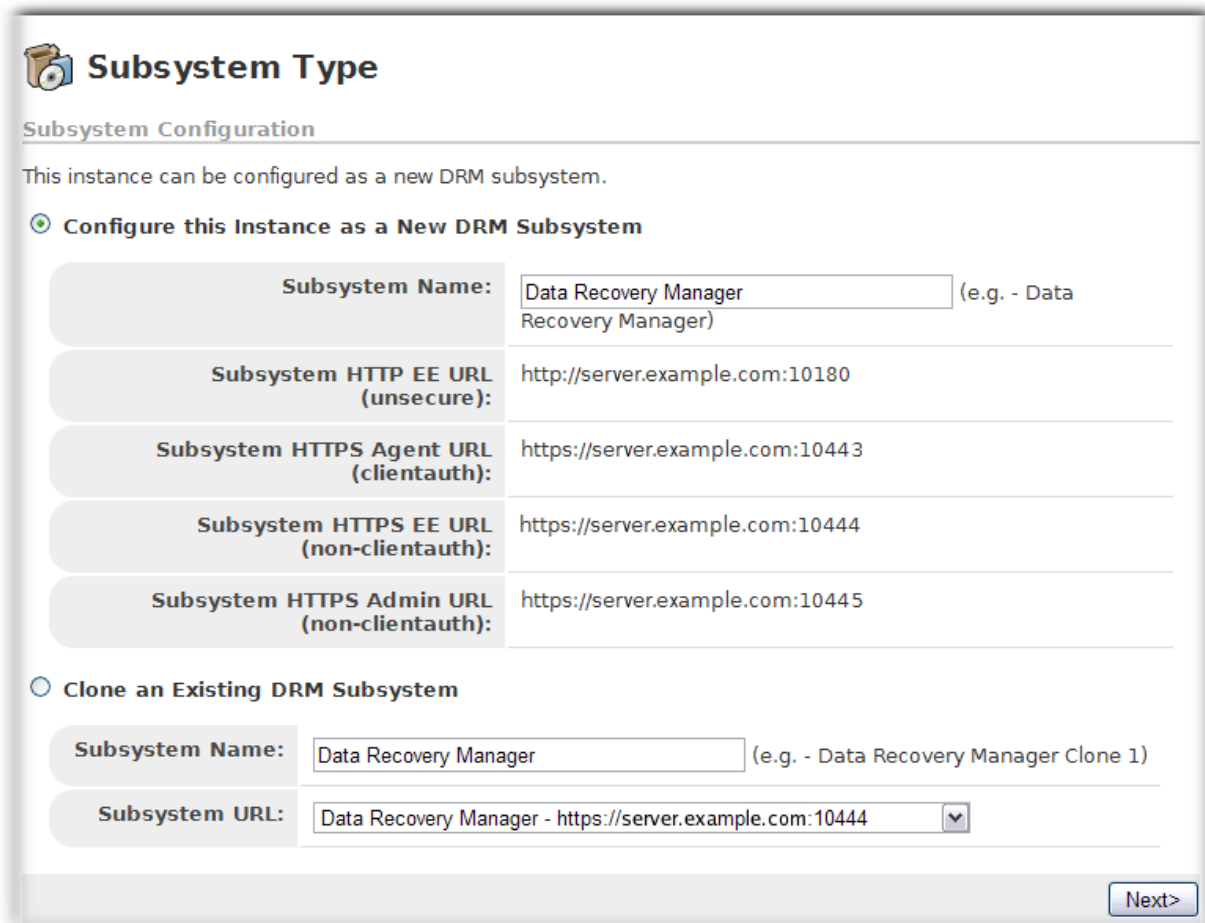
Security Domain (Example Domain) Login

The Enterprise KRA Administrator will register this KRA Subsystem located at `example.com` under this Security Domain located at `example.com`. The credential information will be provided to the Security Domain for authentication.

Uid:

Password:

4. Enter a name for the new instance.



Subsystem Type

Subsystem Configuration

This instance can be configured as a new DRM subsystem.

☒ **Configure this Instance as a New DRM Subsystem**

| | |
|---------------------------------------------|-----------------------------------------------------------------------------------|
| Subsystem Name: | <input type="text" value="Data Recovery Manager"/> (e.g. - Data Recovery Manager) |
| Subsystem HTTP EE URL (unsecure): | <input type="text" value="http://server.example.com:10180"/> |
| Subsystem HTTPS Agent URL (clientauth): | <input type="text" value="https://server.example.com:10443"/> |
| Subsystem HTTPS EE URL (non-clientauth): | <input type="text" value="https://server.example.com:10444"/> |
| Subsystem HTTPS Admin URL (non-clientauth): | <input type="text" value="https://server.example.com:10445"/> |

☐ **Clone an Existing DRM Subsystem**

| | |
|-----------------|-------------------------------------------------------------------------------------------|
| Subsystem Name: | <input type="text" value="Data Recovery Manager"/> (e.g. - Data Recovery Manager Clone 1) |
| Subsystem URL: | <input type="text" value="Data Recovery Manager - https://server.example.com:10444"/> ▼ |

Next>

5. Select the CA which will perform the operations processed through the subsystem, such as key archival.



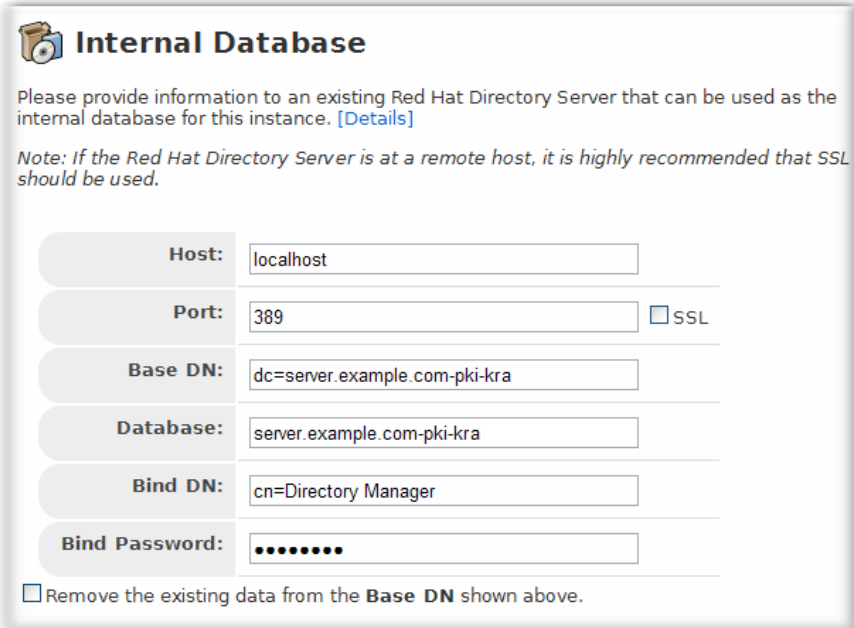
CA Information

A Certificate Authority (CA) is responsible for issuing different kinds of certificates.

URL: Certificate Authority - https://server.example.com:9444

Next>

6. Fill in the information for the LDAP server which will be used for the instance's internal database. This requires connection information for the Directory Server instance, such as the hostname, port number, bind DN (username), and password. This step also creates a database in the Directory Server and a corresponding base directory entry (base DN) to use for the subsystem's entries.



Internal Database

Please provide information to an existing Red Hat Directory Server that can be used as the internal database for this instance. [\[Details\]](#)

Note: If the Red Hat Directory Server is at a remote host, it is highly recommended that SSL should be used.

Host: localhost

Port: 389 ☐ SSL

Base DN: dc=server.example.com-pki-kra

Database: server.example.com-pki-kra

Bind DN: cn=Directory Manager

Bind Password:

☐ Remove the existing data from the **Base DN** shown above.

The hostname can be the fully-qualified domain name or an IPv4 or IPv6 address.



NOTE

One thing that can derail subsystem configuration or function is having services that are unable to connect with each other. If servers that need to communicate with each other are on different servers or networks, when the firewalls and iptables must be configured to give the required access.

If the Red Hat Directory Server instances is on a different server or network than the Certificate System subsystem, then make sure that the Certificate System host's firewall allows access to whatever LDAP port was set in the previous configuration panel. Installation will not complete if iptables is not configured properly. To configure iptables, see the Red Hat Enterprise Linux *Deployment Guide*, such as ["Using iptables."](#) It is also possible to simply turn iptables off.

7. Select the token which will store the Certificate System certificates and keys; a list of detected hardware tokens and databases is given.



IMPORTANT

Any hardware tokens used with the instance must be configured *before* configuring the subsystem instance. If the HSM is not properly configured, it may not be listed in the key stores panel or the instance may not function properly. HSM configuration is described in [Section 2.5.2, “Using Hardware Security Modules with Subsystems”](#).

To determine whether a token is detected by the Certificate System, use the **TokenInfo** tool, as described in [Section 2.5.4, “Detecting Tokens”](#).



Key Store

Two lists of security modules are provided below. The **Supported Security Modules** list consists of both software-based and hardware-based security modules that this PKI solution supports, while the **Other Security Modules** list consists of any other security modules found by this PKI subsystem that are not recognized as one of the supported security modules. [\[Details\]](#)

Supported Security Modules

| Module/Token | Status | Default | Operations |
|----------------------------------------|---------------|---------|-----------------------|
| NSS Internal PKCS #11 Module | Found | | |
| - Internal Key Storage Token | Logged In | | |
| nCipher's nFast Token Hardware Module | Found | | |
| - accelerator | Not logged In | | Login |
| - nethsm | Not logged In | | Login |
| SafeNet's LunaSA Token Hardware Module | Found | | |
| - lunasa1-ca | Not logged In | | Login |
| - lunasa2-ca | Not logged In | | Login |

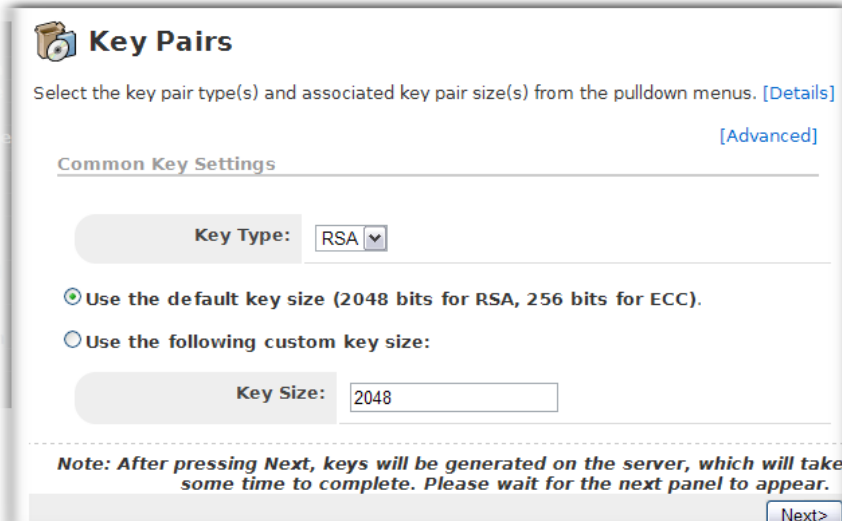
Other Security Modules

The security modules listed below are modules found by the server but not recognized as one of the supported modules. If the user believes that any listed modules below should have been supported, please check the "CS.cfg" configuration file to see if there is a name mismatch and adjust this accordingly.

The Certificate System automatically discovers Safenet's LunaSA and nCipher's netHSM hardware security modules. The discovery process assumes that the client software installations for these modules are local to the Certificate System subsystem and are in the following locations:

- » LunaSA: `/usr/lunasa/lib/libCryptoki2.so`
- » nCipher: `/opt/nfast/toolkits/pkcs11/libcknfast.so`

8. Set the key size. The default RSA key size is 2048.



Key Pairs

Select the key pair type(s) and associated key pair size(s) from the pulldown menus. [\[Details\]](#)

[\[Advanced\]](#)

Common Key Settings

Key Type: RSA ▼

☒ Use the default key size (2048 bits for RSA, 256 bits for ECC).

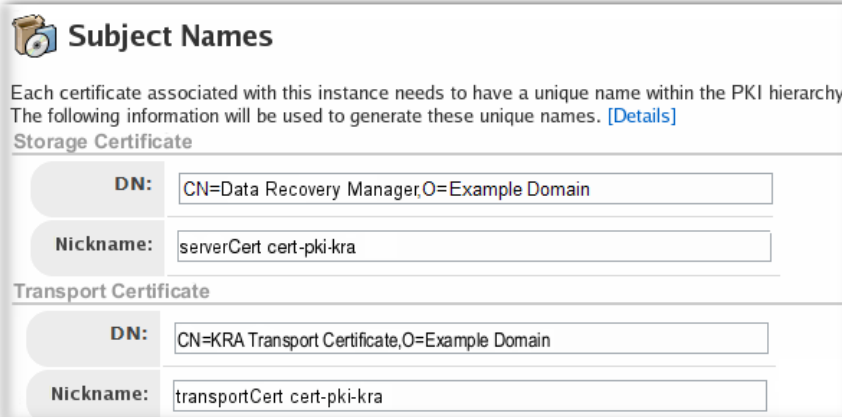
☐ Use the following custom key size:

Key Size: 2048

Note: After pressing Next, keys will be generated on the server, which will take some time to complete. Please wait for the next panel to appear.

[Next>](#)

9. Optionally, change subject names to the listed certificates.



Subject Names

Each certificate associated with this instance needs to have a unique name within the PKI hierarchy. The following information will be used to generate these unique names. [\[Details\]](#)

Storage Certificate

DN: CN=Data Recovery Manager,O=Example Domain

Nickname: serverCert cert-pki-kra

Transport Certificate

DN: CN=KRA Transport Certificate,O=Example Domain

Nickname: transportCert cert-pki-kra



NOTE

Certificate nicknames must be unique, and changing the default nicknames is one way to ensure that. Having unique certificate nicknames is vital for using an HSM, since any nickname conflicts (even for subsystems on different servers) will cause configuration to fail.

10. The next panels generate and show certificate requests, certificates, and key pairs.


Requests and Certificates

A certificate signing request (CSR) contains a public key and is an unsigned copy of the certificate.

If a given CSR has been successfully signed by a CA, then the certificate will be designated below by a certificate icon labeled Certificate Generated Successfully.

However, if a given CSR contains an **action required** label under its certificate icon, then those requests must be manually submitted to a CA for certificate generation. [\[Details\]](#)

Press the [Next] button once all certificates have been generated successfully.
CN=example,O=Example Domain




Certificate Generated Successfully

[View Certificate Request \(CSR\)](#)

[View Certificate in Base64-Encoding](#)

[View Certificate Pretty Print](#)

CN=KRA Transport Certificate,O=Example Domain




Certificate Generated Successfully

[View Certificate Request \(CSR\)](#)

[View Certificate in Base64-Encoding](#)

[View Certificate Pretty Print](#)

CN=KRA Storage Certificate,O=Example Domain



Certificate Generated Successfully

[View Certificate Request \(CSR\)](#)

If an external CA is used to issue the certificates, configuration cannot go forward until they are received from the CA. When they are issued, paste the certificates into this panel to add them to the subsystem database, and then proceed with the installation. Click **Apply** to view the certificates as they are imported.

11. Provide the information for the new subsystem administrator.

Administrator

The administrator is the privileged user who manages this subsystem. Please enter the following relevant information. A certificate request will be automatically generated and submitted. The administrator entry will be created in the internal database and his certificate will be imported into this browser automatically in the next panel. This administrator certificate is used to access the agent interface of this subsystem.

| | |
|-------------------|--------------------------------------------------------------------|
| UID: | <input type="text" value="admin"/> |
| Name: | <input type="text" value="DRM Administrator of Instance pki-kra"/> |
| Email: | <input type="text" value="email@example.com"/> |
| Password: | <input type="password" value="*****"/> |
| Password (Again): | <input type="password" value="*****"/> |

12. Click **Next** through the remaining panels to import the agent certificate into the browser and complete the configuration.
13. When the configuration is complete, restart the subsystem.

```
service pki-kra restart
```



IMPORTANT

The new instance is not active until it is restarted, and weird behaviors can occur if you try to use the instance without restarting it first.

3.6. Configuring a TPS

Subsystem configuration is done by accessing a unique web-based configuration page for the instance. The only supported web browser for subsystem configuration is Mozilla Firefox.



IMPORTANT

Before the TPS can be set up, a Certificate System CA and TKS must be installed and configured. If you want to enable server-side key generation, then the DRM must also be installed and configured. The TPS configuration wizard automatically establishes relationships between the CA, TKS, and DRM and the TPS, so specific CA, TKS, and DRM instances must be available to the TPS.

1. Install the subsystem packages. For example:

```
yum install pki-tps
```

Once the packages are installed, then the installer automatically launches the **pkicreate** script to create the default subsystem instance automatically. A URL to access the new instance is printed to the screen which gives the subsystem instances hostname, port, and a login PIN to access the configuration wizard.

2. Open the configuration wizard using the URL returned by the package installation. For example:

```
http://server.example.com:7888/tps/admin/console/config/login?
pin=kI7E1MBYNIUCPJ6RKHmH
```

Alternatively, log into the setup wizard through admin link on the services page and supply the **preop.pin** value from the **/var/lib/pki-tps/conf/CS.cfg** file when prompted.

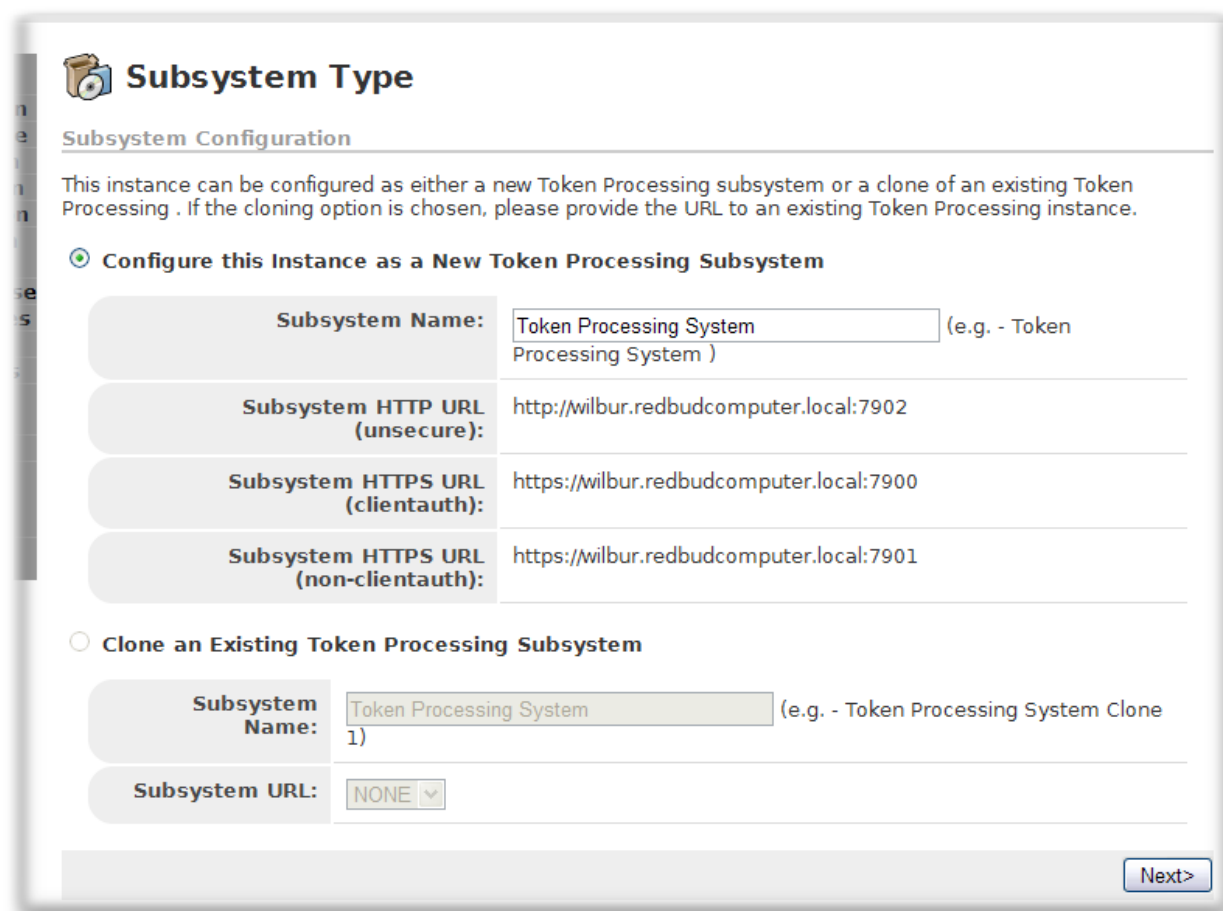
```
http://server.example.com:7888/tps/services
```

3. Join an existing security domain by entering the CA information. This URL can be identified by running **service pki-ca status** on the CA's host; the security domain URL is returned with the other configuration settings. For example:

```
https://server.example.com:9445
```

When the CA is successfully contacted, then supply the admin username and password for the CA so that it can be properly accessed.

4. Enter a name for the new instance.



Subsystem Type

Subsystem Configuration

This instance can be configured as either a new Token Processing subsystem or a clone of an existing Token Processing . If the cloning option is chosen, please provide the URL to an existing Token Processing instance.

☒ **Configure this Instance as a New Token Processing Subsystem**

| | |
|----------------------------------------------|-----------------------------------------------------------|
| Subsystem Name: | Token Processing System (e.g. - Token Processing System) |
| Subsystem HTTP URL (unsecure): | http://wilbur.redbudcomputer.local:7902 |
| Subsystem HTTPS URL (clientauth): | https://wilbur.redbudcomputer.local:7900 |
| Subsystem HTTPS URL (non-clientauth): | https://wilbur.redbudcomputer.local:7901 |

☐ **Clone an Existing Token Processing Subsystem**

| | |
|------------------------|------------------------------------------------------------------|
| Subsystem Name: | Token Processing System (e.g. - Token Processing System Clone 1) |
| Subsystem URL: | NONE |

Next>

5. Select the CA which will issue, renew, and revoke certificates for token operations requested through the TPS subsystem.



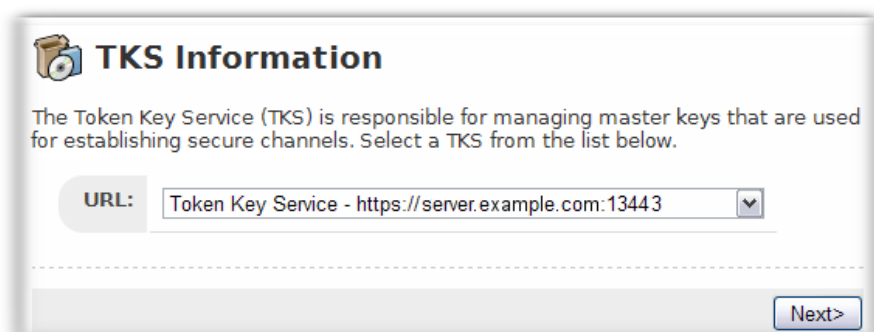
CA Information

A Certificate Authority (CA) is responsible for issuing different kinds of certificates.

URL: Certificate Authority - https://server.example.com:9444

Next>

6. Supply information about the TKS which will manage the TPS keys. Select the TKS from the drop-down menu of TKS subsystems within the security domain.



TKS Information

The Token Key Service (TKS) is responsible for managing master keys that are used for establishing secure channels. Select a TKS from the list below.

URL: Token Key Service - https://server.example.com:13443

Next>

7. There is an option for server-side key generation for tokens enrolled through the TPS. If server-side key generation is selected, supply information about the DRM which will generate keys and archive encryption keys. Key and certificate recovery is initiated automatically through the TPS,

which is a DRM agent. Select the DRM from the drop-down menu of DRM subsystems within the security domain.

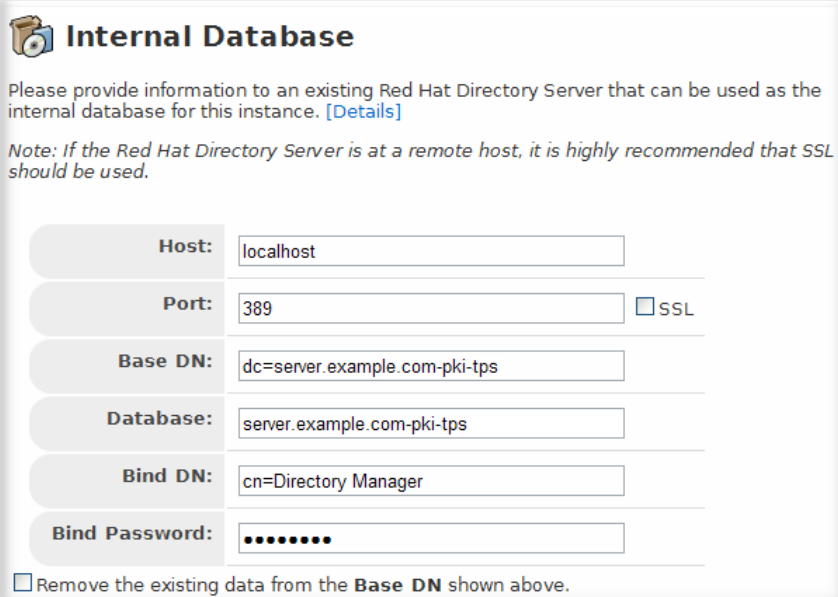
The hostname for the DRM can be the fully-qualified domain name or an IPv4 or IPv6 address.

8. Fill in the Directory Server authentication directory. This directory is used by the TPS to authenticate users which access the Enterprise Security Client and as an additional database for certificates and keys.

This Directory Server instance is *not* the same Directory Server instance used as the TPS's internal database. This is a general user directory, which may be accessed by other applications or users. The TPS's internal database is used exclusively by the TPS and is created on the fly as the TPS is configured.

The hostname of the LDAP server can be the fully-qualified domain name or an IPv4 or IPv6 address.

9. Fill in the information for the LDAP server which will be used for the instance's internal database. This requires connection information for the Directory Server instance, such as the hostname, port number, bind DN (username), and password. This step also creates a database in the Directory Server and a corresponding base directory entry (base DN) to use for the subsystem's entries.



Internal Database

Please provide information to an existing Red Hat Directory Server that can be used as the internal database for this instance. [\[Details\]](#)

Note: If the Red Hat Directory Server is at a remote host, it is highly recommended that SSL should be used.

| | |
|----------------|----------------------------------|
| Host: | localhost |
| Port: | 389 <input type="checkbox"/> SSL |
| Base DN: | dc=server.example.com-pki-tps |
| Database: | server.example.com-pki-tps |
| Bind DN: | cn=Directory Manager |
| Bind Password: | •••••••• |

☐ Remove the existing data from the **Base DN** shown above.

The hostname can be the fully-qualified domain name or an IPv4 or IPv6 address.



NOTE

One thing that can derail subsystem configuration or function is having services that are unable to connect with each other. If servers that need to communicate with each other are on different servers or networks, when the firewalls and iptables must be configured to give the required access.

If the Red Hat Directory Server instances is on a different server or network than the Certificate System subsystem, then make sure that the Certificate System host's firewall allows access to whatever LDAP port was set in the previous configuration panel. Installation will not complete if iptables is not configured properly. To configure iptables, see the Red Hat Enterprise Linux *Deployment Guide*, such as ["Using iptables."](#) It is also possible to simply turn iptables off.

10. Select the token which will store the Certificate System certificates and keys; a list of detected hardware tokens and databases is given.



IMPORTANT

Any hardware tokens used with the instance must be configured *before* configuring the subsystem instance. If the HSM is not properly configured, it may not be listed in the key stores panel or the instance may not function properly. HSM configuration is described in [Section 2.5.2, "Using Hardware Security Modules with Subsystems"](#).

To determine whether a token is detected by the Certificate System, use the **TokenInfo** tool, as described in [Section 2.5.4, "Detecting Tokens"](#).



Key Store

Two lists of security modules are provided below. The **Supported Security Modules** list consists of both software-based and hardware-based security modules that this PKI solution supports, while the **Other Security Modules** list consists of any other security modules found by this PKI subsystem that are not recognized as one of the supported security modules. [\[Details\]](#)

Supported Security Modules

| Module/Token | Status | Default | Operations |
|----------------------------------------|---------------|---------|-----------------------|
| NSS Internal PKCS #11 Module | Found | | |
| - Internal Key Storage Token | Logged In | | |
| nCipher's nFast Token Hardware Module | Found | | |
| - accelerator | Not logged In | | Login |
| - nethsm | Not logged In | | Login |
| SafeNet's LunaSA Token Hardware Module | Found | | |
| - lunasa1-ca | Not logged In | | Login |
| - lunasa2-ca | Not logged In | | Login |

Other Security Modules

The security modules listed below are modules found by the server but not recognized as one of the supported modules. If the user believes that any listed modules below should have been supported, please check the "CS.cfg" configuration file to see if there is a name mismatch and adjust this accordingly.

The Certificate System automatically discovers Safenet's LunaSA and nCipher's netHSM hardware security modules. The discovery process assumes that the client software installations for these modules are local to the Certificate System subsystem and are in the following locations:

- » LunaSA: `/usr/lunasa/lib/libCryptoki2.so`
- » nCipher: `/opt/nfast/toolkits/pkcs11/libcknfast.so`

11. Set the key size.



Key Pairs

Select the key pair type(s) and associated key pair size(s) from the pulldown menus. [\[Details\]](#)

[\[Advanced\]](#)

Common Key Settings

Key Type: RSA ▼

☒ Use the default key size (2048 bits for RSA, 256 bits for ECC).

☐ Use the following custom key size:

Key Size:

Note: After pressing Next, keys will be generated on the server, which will take some time to complete. Please wait for the next panel to appear.

[Next>](#)

12. Optionally, change subject names to the listed certificates.

Subject Names

Each certificate associated with this instance needs to have a unique name within the PKI hierarchy. The following information will be used to generate these unique names. [\[Details\]](#)

SSL Server Certificate

DN: CN=Token Processing System,O=Example Domain

Nickname: serverCert cert-pki-tps

Subsystem Certificate

DN: CN=TPS Subsystem Certificate,O=Example Domain

Nickname: subsystemCert cert-pki-tps



NOTE

Certificate nicknames must be unique, and changing the default nicknames is one way to ensure that.

Having unique certificate nicknames is vital for using an HSM, since any nickname conflicts (even for subsystems on different servers) will cause configuration to fail.

- The next panels generate and show certificate requests, certificates, and key pairs.

Requests and Certificates

A certificate signing request (CSR) contains a public key and is an unsigned copy of the certificate.

If a given CSR has been successfully signed by a CA, then the certificate will be designated below by a certificate icon labeled Certificate Generated Successfully.

However, if a given CSR contains an **action required** label under its certificate icon, then those requests must be manually submitted to a CA for certificate generation. [\[Details\]](#)

Press the [Next] button once all certificates have been generated successfully.

CN=example,O=Example Domain



Certificate Generated
Successfully

[View Certificate Request \(CSR\)](#)

[View Certificate in Base64-Encoding](#)

[View Certificate Pretty Print](#)

CN=TPS Subsystem Certificate,O=Example Domain



Certificate Generated
Successfully

[View Certificate Request \(CSR\)](#)

[View Certificate in Base64-Encoding](#)

[View Certificate Pretty Print](#)

Apply


<Back

Next>

If an external CA is used to issue the certificates, configuration cannot go forward until they are

received from the CA. When they are issued, paste the certificates into this panel to add them to the TPS database, and then proceed with the installation. Click **Apply** to view the certificates as they are imported.

14. Provide the information for the new subsystem administrator.

 **Administrator**

The administrator is the privileged user who manages this subsystem. Please enter the following relevant information. A certificate request will be automatically generated and submitted. The administrator entry will be created in the internal database and his certificate will be imported into this browser automatically in the next panel. This administrator certificate is used to access the agent interface of this subsystem.

| | |
|-------------------|--------------------------------------------------------------------|
| UID: | <input type="text" value="admin"/> |
| Name: | <input type="text" value="TPS Administrator of Instance pki-tps"/> |
| Email: | <input type="text" value="email@example.com"/> |
| Password: | <input type="password" value="*****"/> |
| Password (Again): | <input type="password" value="*****"/> |

15. Click **Next** through the remaining panels to import the agent certificate into the browser and complete the configuration.
16. When the configuration is complete, restart the subsystem.

```
service pki-tps restart
```



IMPORTANT

The new instance is not active until it is restarted, and weird behaviors can occur if you try to use the instance without restarting it first.

Chapter 4. Additional Installation Options

The Certificate System default installation, and all subsequent instances created with **pkicreate**, make certain assumptions about the instances being installed, such as the default signing algorithm to use for CA signing certificates and whether to allow IPv6 addresses for hosts.

This chapter describes additional configuration options that impact the installation and configuration for new instances, so many of these procedures occur before the instance is created.

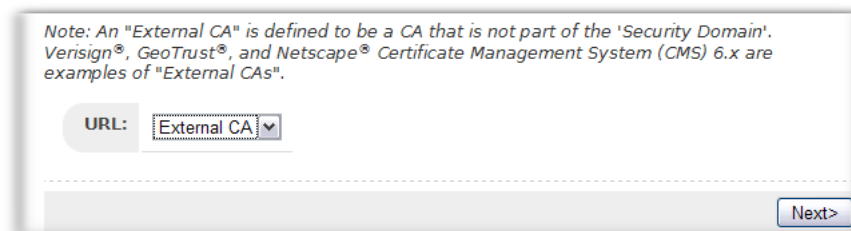
4.1. Requesting Subsystem Certificates from an External CA

Most of the time, it is easier and simpler to have a CA within your PKI be the root CA, since this affords a lot of flexibility for defining the rules and settings of the PKI deployment. However, for public-facing networks, this can be a difficult thing to implement, because web administrators have to find some way to propagate and update CA certificate chains to their clients so that any site is trusted. For this reason, people often use public CAs, hosted by companies like VeriSign or Thawte, to issue CA signing certificates and make all of their CAs subordinate to the public CA. This is one of the planning considerations covered in the *Certificate System Deployment Guide*.

All subsystem certificates can be submitted to an external CA when the subsystem is configured. When the certificates are generated from a CA outside the Certificate System deployment (or from a Certificate System CA in a different security domain), then the configuration process does not occur in one sitting. The configuration process is on hold until the certificates can be retrieved. Aside from that delay, the process is more or less the same as in [Chapter 3, Installation and Configuration](#).

1. Install the subsystem packages, and open the configuration URL.
2. Join a security domain. For a CA, it is also possible to create a new security domain.
3. Set the subsystem information, like its name.
4. For a CA, set the CA to be a subordinate CA. This is required in order to submit CA subsystem certificates to an external CA; making a root CA automatically generates self-signed certificates.
5. Set up the internal database.
6. Select the key store, and generate the key pairs.
7. Set the subsystem certificate names; you can set these to whatever you want, but make sure that they conform to any requirements that the external CA sets for the subject names of certificates.

At the bottom of the screen is the list of CAs which are available to accept the submitted certificate requests. Choose **External CA** from the drop-down menu.



8. In the **Requests and Certificates** panel, the CA signing certificate is listed, with an **action required** label. Once that certificate is generated, the other certificates for the CA will be automatically generated.

For other subsystems, each subsystem certificate has an **action required** label and must be submitted to the external CA.



Requests and Certificates

A certificate signing request (CSR) contains a public key and is an unsigned copy of the certificate.

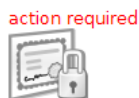
If a given CSR has been successfully signed by a CA, then the certificate will be designated below by a certificate icon labeled Certificate Generated Successfully.

However, if a given CSR contains an **action required** label under its certificate icon, then those requests must be *manually* submitted to a CA for certificate generation.

Press the [Apply] button after certificates and chains are pasted in.

Press the [Next] button once all certificates have been generated successfully.

CN=Certificate Authority,O=New Domain



Step 1: Copy the Certificate Request (CSR) to enroll at an external CA

Step 2: Import the PKCS #7 Certificate Chain (optional if the certificate already contains the chain)

Step 3: Paste in the Base64-encoded Certificate after enrollment at an external CA (NOTE: this text box does not accept PKCS #7 certificate chains)

CN=OCSP Signing Certificate,O=New Domain



certificate will be
generated
internally

- Click the **Step 1: Copy the certificate request** link, and copy the certificate request to a file or the clipboard.

Copy the following Certificate Request (CSR) and paste it in the external CA enrollment page for enrollment

```
-----BEGIN CERTIFICATE REQUEST-----
MIICejCCAWICAQAwNTETMBEGA1UEChMKTMV3IERvbWVpbjEeMBwGA1UEAxMVQ2VydGhmaWNhdGUG
QXV0aG9yaXR5SMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAnSpOsyu0KQ7Uo/6dBmi
CDsRxFggF/1Np1cgj3Pab7752uMxPSY4PnOOT5BKQMxfI+DNLysu0BEmtVF/6SHBYVHr90jDUg8
CAY/dK3U9Yq1g0uxSxI8SGCULd5MT2BUAKjeNfOGfKvttSaDECD257fAni5m1JgIuF4PgaObdztn
ThzjZ1yHPi5y509+091x6CtX1Mb0geZP217eMeostdeLFvjv9XGFWBkVCpVQ2SA9v034HHjeaP8M
mHXZa74i3m1ZggJL4MANSLOyIo2zrmQ1cWJ/xV6XN7dgKOiI7YzMd1/ttCTYqL3efBzr5mrczaSg
fA9UpG19BrVnDqCp1QIDAQABAAwDQYJKoZIhvcNAQEEBQADggEBAGMHqWRo6zmy311BIqrL7MEd
w5fFpUpPEA67K2BnpW0AXwjudEGV/DSRzcnR/G/xD1xy3hQhQa2cvXXafft+OswGUQYCiHMrb/
PrT5ip2ltpBxdwotY+uHjFuCrkCP9nzH93SdCpdtXP1a6sbmTdvYG11sNy+KAXPeL73MSn+wzW/j
nxLMKC7/Oh/m89Lw7T/wXL8P349BZsYPqS+EW83907Y4LDa1Wtx0Ru611LbRhjhBQZRh44S2S9fI
b0r6owp15fISC+Q1fPZp8jGwj1F4q2rDrFL8ER1GKgCWODp1dk2OXwp9SQjxbGa4yBh1Lq9wJqS
BUeQHycrIC7Hzw0=
-----END CERTIFICATE REQUEST-----
```

- Submit the request to the external CA. Leave the browser with the configuration wizard open as you wait for the certificates to be generated, so that it is easier to pick up the process.
- Click **Step 2**, and import the certificate chain for the issuing CA. This certificate chain must *not* contain the leaf certificate (the certificate being requested).
- After retrieving the issued certificates, click the **Step 3: Paste in the base 64-encoded certificate** link, and paste in the new certificate (this should be only the new certificate, not a certificate chain).

Certificate System

Copy the resulting base64-encoded certificate (NOTE: PKCS #7 not accepted) into the text box below and press 'X'

...paste certificate here...

- For the CA, this only has to be done for the signing certificate. For the other subsystems, repeat

steps [9](#), [10](#), and [12](#) for each certificate.

When all the certificates have been submitted, click **Next** to resume the configuration process.

14. Export the keys for the certificates, if the subsystem will be cloned.
15. Create the administrator user.
16. When the configuration is done, restart the subsystem.

```
service subsystem_name restart
```

4.2. Installing a CA with ECC Enabled

Elliptic curve cryptography (ECC) is much more secure than the more common RSA-style encryption, which allows it to use much shorter key lengths and makes it faster to generate certificates. CAs which are ECC-enabled can issue both RSA and ECC certificates, using their ECC signing certificate.

Certificate System does not include a module natively to enable ECC, but it is possible to load and use a third-party PKCS #11 module with ECC-enabled.

To use the ECC module, it must be loaded before the subsystem instance is configured.



IMPORTANT

Third-party ECC modules must have an SELinux policy configured for them, or SELinux needs to be changed from **enforcing** mode to **permissive** mode to allow the module to function. Otherwise, any subsystem operations which require the ECC module will fail.

4.2.1. Loading a Third-Party ECC Module

1. Copy the third-party module to a common directory, like **/usr/lib** for 32-bit systems or **/usr/lib64** for 64-bit systems.
2. Create a new CA instance by running **pkicreate**, but do *not* go through the configuration wizard.
3. Stop the CA.

```
service pki-ca stop
```

4. The CA runs as the **pkiuser** user. As **root**, create a home directory for **pkiuser**.

```
/usr/sbin/usermod --home /usr/share/pki/pkiuser pkiuser
cd /usr/share/pki
mkdir pkiuser
HOME=/usr/share/pki/pkiuser
export HOME
```

5. Install the third-party module in the CA's security databases so it is available for the configuration.

```
cd /var/lib/pki-ca/alias

modutil -dbdir . -nocertdb -add THIRD_PARTY_MODULE -libfile
/usr/lib/libYourNewModule.so
```

This creates a directory called **.THIRD_PARTY_MODULE** in the new home directory created for

root (the new **pkiuser** home directory). For example, if the module's name is **EccForPki**, then the directory is named **.EccForPki/**

- Using **modutil**, set the password for the new ECC module token.

```
modutil -dbdir . -nocertdb -change pw "THIRD_PARTY_MODULE_TOKEN"
```

- Change the ownership of the new home directory from **root** to **pkiuser**.

```
cd /usr/share/pki
chown -R pkiuser:pkiuser pkiuser
```

- Add the password for the ECC token to the CA's password file.

```
vim /etc/pki-ca/password.conf

hardware-THIRD_PARTY_MODULE_TOKEN=secret
```

The **hardware-** prefix is required.

- Edit the CA configuration and add a line to require signature verification.

```
ca.requestVerify.token=THIRD_PARTY_MODULE_TOKEN
```

- Start the CA.

```
service pki-ca start
```

- Continue with the CA configuration, with two important configuration settings:

- ▶ In the **Key Store** panel, the ECC module should be listed as an available token. Select that module for the key store.
- ▶ In the **Key Pairs** panel, ECC should be listed as an option to use to generate the keys used for the CA's certificates. Select the ECC key type.

- After completing the configuration for the CA, try to log into the CA console.

```
pkiconsole https://server.example.com:9445/ca
```

This fails, because the console is not yet configured to run with ECC enabled. However, this does create the security databases for the console, so the ECC module can be loaded.

- Load the ECC module into the console security databases.

```
cd ~/.redhat-idm-console/

modutil -dbdir . -nocertdb -add THIRD_PARTY_MODULE -libfile
/usr/lib/libYourNewModule.so
```

Now, logging into the console succeeds.

4.2.2. Loading the Certicom ECC Module

Certicom's ECC module has a slightly different configuration process than the procedure for loading a general ECC module.

- Copy the third-party libraries to a common directory, like **/usr/lib** for 32-bit systems or **/usr/lib64** for 64-bit systems.

There are two library files for the Certicom ECC modules, **libsbcpgse.so** and **libsbpgse2.so**.

2. Cache the recent shared libraries.

```
ldconfig
```

3. Install the CA, but do *not* go through the configuration wizard.
4. Stop the CA.

```
service pki-ca stop
```

5. The CA runs as the **pkiuser** user. As **root**, create a home directory for **pkiuser**.

```
/usr/sbin/usermod --home /usr/share/pki/pkiuser pkiuser
cd /usr/share/pki
mkdir pkiuser
HOME=/usr/share/pki/pkiuser
export HOME
```

6. Open the subsystem's **alias** directory. For example:

```
cd /var/lib/pki-ca/alias
```

7. Install the third-party module in the CA's security databases so it is available for the configuration.

```
modutil -dbdir . -nocertdb -add certicom -libfile /usr/lib/libsbcpge.so
```

This creates a **.certicom** directory in the new **pkiuser** home directory.

8. Certicom's ECC module includes an **initpin** file; copy this into the new **pkiuser** directory and give it execute permissions. For example:

```
cp /tmp/initpin /usr/share/pki/pkiuser
chmod +x initpin
```

9. Run Certicom's **initpin** file from the **/usr/share/pki/pkiuser** directory. This first prompts for the directory to use for the Certicom token databases; use the **pkiuser** home directory, **/usr/share/pki/pkiuser**. This also prompts to set a password for the module, and then proceed with configuring the module.

```
/usr/share/pki/pkiuser/initpin
```

```
Please enter the directory where the token databases exist or will
be created: /usr/share/pki/pkiuser
```

```
Enter PIN:
```

```
Confirm PIN:
```

```
Security Builder API for PKCS #11 Samples
```

```
    CryptoAes() success
    CryptoArc4() success
    CryptoDes() success
    CryptoDh() success
    CryptoDsa() success
    CryptoEcdh() success
    CryptoEcdsa() success
    CryptoEcmqv() success
    CryptoPkcs1Enc() success
    CryptoPkcs1Sig() success
    CryptoRsaEnc() success
    CryptoRsaSig() success
    CryptoSha1() success
    Token() samples starting
```

```
Slot info for Slot 0
```

```
Desc: FIPS Generic Crypto Services V1.0.1d
```

```
manufacturerID: Certicom Corp.
```

```
flags: 0x1
```

```
        CKF_TOKEN_PRESENT
```

```
hardwareVersion: 1.0
```

```
...
```

10. Edit the **pkiuser**'s home directory so that every file is owned by **pkiuser**.

```
cd /usr/share/pki; chown -R pkiuser:pkiuser pkiuser
```

11. List the Certicom ECC module to make sure it has been properly loaded. The module is in security databases in the subsystem's **alias** directory. For example:

```
modutil -dbdir /var/lib/pki-ca/alias -list certicom
```

12. Add the password for the ECC token to the subsystem's password file. Escape any spaces or special characters in the name. For example:

```
vim /etc/pki-ca/password.conf
```

```
hardware-Certicom\ FIPS\ Cert/Key\ Services=secret
```

The **hardware-** prefix is required.

13. Edit the CA configuration and add a line to require signature verification. In this file, spaces and special characters do not need to be escaped. For example:

```
ca.requestVerify.token=Certicom FIPS Cert/Key Services
```

14. Edit file **dtomcat5-instance** file for the subsystem in the **/usr/bin** directory, and add a line to use the ECC module.

```
umask 00002
NSS_USE_DECODED_CKA_EC_POINT=1
export NSS_USE_DECODED_CKA_EC_POINT
```

15. Start the CA.

```
service pki-ca start
```

16. Continue with the CA configuration, with two important configuration settings:

- In the **Key Store** panel, the ECC module should be listed as an available token. Select that module for the key store.
- In the **Key Pairs** panel, ECC should be listed as an option to use to generate the keys used for the CA's certificates. Select the ECC key type.

17. After completing the configuration, try to log into the subsystem console.

```
pkiconsole https://server.example.com:9445/ca
```

This fails, because the console is not yet configured to run in ECC. However, this does create the security databases for the console, so the ECC module can be loaded.

Load the ECC module into the console security databases.

```
cd ~/.redhat-idm-console/

modutil -dbdir . -nocertdb -add certicom -libfile /usr/lib/libsbcpge.so
```

Now, logging into the console succeeds.

18. The web browser used to access administrative and agent services pages also needs to be configured to support ECC.

- a. Create a user for the browser profile, such as **agent-pki**.
- b. Launch Firefox and create a profile for this user; this automatically creates the required security databases and directory.
- c. Set the **root** home directory to **/home/agent-pki**, and make sure the directory is owned by **root**.

```
chown -R root:root /home/agent-pki
```

- d. Copy the ECC module libraries and **initpin** file to the **/home/agent-pki** directory. All these files should be owned by **root**.
- e. Load the ECC module.

```
modutil -dbdir /home/agent-pki/.mozilla/profile.default -nocertdb -add
certicom -libfile /usr/lib/libsbcpge.so
```

- f. Run the **initpin** file. When prompted, enter the Certicom token database directory, **/usr/share/pki/pkiuser**, and enter the PIN configured for those databases.

```
./initpin
```

- g. Change the ownership of the new user's home directory from **root** to the user. For example:

```
chown -R agent-pki:agent-pki /home/agent-pki
```

- h. In the terminal with the **/home/agent-pki** directory open, export the environment variable that allows ECC support.

```
export NSS_USE_DECODED_CKA_EC_POINT=1
```

- i. Open Firefox again. The Certicom module should be available and you should be able to log into it successfully.
- j. Then, import the agent certificate and root CA certificate or certificate chain into Firefox so that the user profile can access the agent services pages.
19. The **NSS_USE_DECODED_CKA_EC_POINT** environment variable also needs to be set to access the subsystem Java console with an ECC certificate. This can be set in the **.bashrc** file for the user who uses the console. For example:

```
vim /home/jsmith/.bashrc

# User specific aliases and functions
NSS_USE_DECODED_CKA_EC_POINT=1
export NSS_USE_DECODED_CKA_EC_POINT
```

4.3. Changing the Hashing Algorithm Used for Subsystem Keys

When a CA is installed, along with determining the key type and size, the hashing algorithm for the key pair is set. However, for other subsystems, the hashing algorithm is not configurable, so they use whatever the default is for the CA which issues their certificates.

Instead of using the CA's hashing algorithm, it is possible to edit the profiles used to generate the subsystem certificates; then, the configuration wizard will use whatever hashing algorithm is in the profile instead of the one used by the CA.



NOTE

This is true for subordinate CAs as well as other subsystems. While some of the certificates for a sub CA are generated locally — and therefore can take a user-defined hashing algorithm for their keys in the configuration wizard — other certificates for the sub CA (like its signing certificate) are generated by another CA and default to that CA's key hashing algorithm.

To assign the hashing algorithm to the certificate, add this line to the profile in the CA's profile directory, such as **/var/lib/instance_name/profiles/ca**:

```
default.params.signingAlg=hashing_alg
```



TIP

Editing certificate profiles is covered in the *Administrator's Guide*.

Each of the subsystem certificate profiles can be edited:

- `caInternalAuthOCSPCert.cfg`
- `caInternalAuthTransportCert.cfg`
- `caInternalAuthAuditSigningCert.cfg`
- `caInternalAuthServerCert.cfg`
- `caInternalAuthDRMstorageCert.cfg`
- `caInternalAuthSubsystemCert.cfg`

The hashing algorithms that are available depend on whether RSA or ECC is selected as the key type. For RSA, the available algorithms are as follows:

- SHA256withRSA (the default)
- SHA1withRSA
- SHA256withRSA
- SHA512withRSA
- MD5withRSA
- MD2withRSA

For ECC:

- SHA256withEC (the default)
- SHA1withEC
- SHA384withEC
- SHA512withEC

4.4. Enabling IPv6 for a Subsystem

Certificate System automatically configures and manages connections between subsystems. Every subsystem must interact with a CA as members of a security domain and to perform their PKI operations.

For these connections, Certificate System subsystems can be recognized by their host's fully-qualified domain name or an IP address. By default, Certificate System resolves IPv4 addresses and hostnames automatically, but Certificate System can also use IPv6 for their connections. IPv6 is supported for all server connections: to other subsystems, to the administrative console (**pkiconsole**), or through command-line scripts such as **tpscclient**.

```
op=var_set name=ca_host value=IPv6 address
```

If a host has both an IPv4 address and an IPv6 name, then an environment variable can be set *before* the Certificate System packages are installed so that Certificate System setup scripts will recognize the IPv6 name.

1. If the default (first) instances should use IPv6, set the environment variable to keep **pkicreate** from running.

```
export DONT_RUN_PKICREATE=1
```

2. Install the Red Hat Certificate System packages.
3. Set the IPv4 and IPv6 addresses in the `/etc/hosts` file. For example:

```
vim /etc/hosts

192.0.0.0      server.example.com IPv4 address
3ffe:1234:2222:2000:202:55ff:fe67:f527      server6.example.com IPv6
address
```

- Then, export the environment variable to use the IPv6 address for the server. For example:

```
export PKI_HOSTNAME=server6.example.com
```

- Unset the **DONT_RUN_PKICREATE** variable so that the new instances can be created.

```
export DONT_RUN_PKICREATE=0
```

- Run **pkicreate** to create the new instance. The values for the server hostname in the **CS.cfg** file will be set to the IPv6 address.

4.5. Configuring Separate RA Instances

When an RA is installed or created, it is automatically added to a default Registration Managers Group on the CA. This means that all RA managers belong to the same group, by default.

However, a particular site might require more than one RA instance, each having its own set of RA agents. If the site policy disallows cross-management between the RA instances, then extra configuration is needed to create separate RA groups.

- Install and configure the *first* RA instance.
- Add the new RA group to the Certificate Manager.
 - Start the Console. For example:

```
pkiconsole https://server.example.com:9445/ca
```

- Click **Users and Groups**, and then click **Groups**.
 - Click **Add** to open the **Edit Group Information** dialog box.
 - Enter the group name and description, such as **Registration Manager2 Agents**.
 - Click **OK**.
- Add the new RA authentication instance to the CA:
 - Open the CA configuration directory, and edit the **CS.cfg** file

```
cd /etc/pki-ca

vi CS.cfg
```

- Search for the string **raCertAuth**.
- Copy those lines for the first RA instance, paste them, and edit them for the second RA instance's information. For example:

```
auths.instance.raCertAuth.agentGroup=Registration Manager Agents
auths.instance.raCertAuth.pluginName=AgentCertAuth
auths.instance.ra2CertAuth.agentGroup=Registration Manager2 Agents

auths.instance.ra2CertAuth.pluginName=AgentCertAuth
```

4. Add the new RA user enrollment profile to the Certificate Manager's certificate profiles list to utilize the new RA authentication instance.

- a. Open the CA profiles directory.

```
cd /var/lib/pki-ca/profiles/ca
```

- b. Copy the current RA profile to create the new profile. For example:

```
cp caDualRAUserCert.cfg caDualRA2userCert.cfg
```

- c. Edit the new file to contain the second RA instance's information. Change **raCertAuth** to **ra2CertAuth**.

5. Open the CA configuration directory, and edit the **CS.cfg** file.

```
cd /var/lib/pki-ca/conf
vi CS.cfg
```

- a. Add **caDualRA2userCert** to the profiles list. For example:

```
profile.list=...[snip]...caRAserverCert,caRA2userCert
```

Make sure to use a comma to separate the entries.

- b. Search for the lines for the **caDualRAUserCert** profile configuration, copy them, and edit them for the second RA instance's information.

```
profile.caDualRAUserCert.class_id=caEnrollImpl
profile.caDualRAUserCert.config=/var/lib/pki-
ca/profiles/ca/caDualRAUserCert.cfg
profile.caDualRA2userCert.class_id=caEnrollImpl
profile.caDualRA2userCert.config=/var/lib/pki-
ca/profiles/ca/caDualRA2userCert.cfg
```

6. Add a new URI mapping to allow the new RA agent to be registered in the new RA group.

- a. Open the CA web applications directory, and edit the **web.xml** file:

```
cd /var/lib/pki-ca/webapps/ca/WEB-INF
vi web.xml
```

- b. At about line 288 in the **web.xml** file is the **servlet** setting for the first RA's user. Copy the entire entry, including the opening and closing **<servlet>** tags, and edit the information to match the second RA's user. For example:

```

<servlet>
  <servlet-name> caRegisterRa2User </servlet-name>
  <servlet-class> com.netscape.cms.servlet.csadmin.RegisterUser
</servlet-class>
    <init-param><param-name> GetClientCert </param-name>
      <param-value> false </param-value> </init-
param>
    <init-param><param-name> authority </param-name>
      <param-value> ca </param-value> </init-
param>
    <init-param><param-name> ID </param-name>
      <param-value> caRegisterRaUser </param-value>
</init-param>
    <init-param><param-name> AuthMgr </param-name>
      <param-value> TokenAuth </param-value> </init-param>
    <init-param><param-name> GroupName </param-name>
      <param-value> Registration Manager2 Agents </param-
value> </init-param>
    <init-param><param-name> AuthzMgr </param-name>
      <param-value> BasicAclAuthz </param-value> </init-
param>
    <init-param><param-name> resourceID </param-name>
      <param-value> certServer.ca.registerUser </param-
value> </init-param>
</servlet>

```

- c. At about line 2510 in the **web.xml** file is the **servlet-mapping** setting for the first RA's user mapping. Copy the entire entry, including the opening and closing **<servlet-mapping>** tags, and edit the information to match the second RA's user. For example:

```

<servlet-mapping>
  <servlet-name> caRegisterRa2User </servlet-name>
  <url-pattern> /admin/ca/registerRa2User </url-pattern>
</servlet-mapping>

```

7. Restart the CA. For example:

```
service pki-ca restart
```

8. Create the new RA instance using the **pkicreate**.

```
pkicreate -pki_instance_root=/var/lib -subsystem_type=ra -
pki_instance_name=pki-ra2 -secure_port=12899 -unsecure_port=12898 -verbose -
user=pkiuser -group=pkiuser
```

9. Open the configuration file for the new RA instance, and edit its parameters to reflect the second RA instance information.

```
cd /var/lib/pki-ra2/conf/
vi CS.cfg
```

10. Change the **registerRaUser** setting to **registerRa2User**.

```
conn.ca1.servlet.addagent=/ca/admin/ca/registerRa2User
```

11. Change the **caDualRAUserCert** setting to **caDualRA2userCert**.

```
request.renewal.approve_request.0.profileId=caDualRAuser2Cert
...
request.user.approve_request.0.profileId=caDualRA2userCert
```

12. Restart the new RA instance. For example:

```
# service pki-ra2 restart
```

13. A URL was generated at the end of the **pkicreate** command; go to that URL to configure the second RA. For example:

```
http://server.example.com:12898/ra/admin/console/config/login?
pin=bFyAk9nWPfgLZXffRBT9
```

14. When the new RA is completely configured, restart the instance.

```
# service pki-ra2 restart
```

Chapter 5. Creating Additional Subsystem Instances

The number of subsystems that you have is flexible. There can be a single instance, there can be multiple instances on the same machine, or there can be multiple instances on multiple servers.

Creating additional subsystem instances is similar to installing and configuring the default instances; there is a script to run to create a basic installation and then an HTML-based configuration wizard to complete the setup for the instance.

All additional CA, RA, DRM, OCSP, TKS, and TPS instances are installed by running a special tool, **pkicreate**. After that, they are configured through the HTML-based administration page. For more information on **pkicreate**, see the *Certificate System Command-Line Tools Guide*.



TIP

Existing instances can be duplicated (cloned) for load balancing for heavily trafficked servers and for failover support. Cloning is described in [Chapter 6, Cloning Subsystems](#).

5.1. About pkicreate

Certificate System subsystem instances are created and defined using a script called **pkicreate**. This script is run automatically when the subsystem packages are first installed, and it creates the default subsystem instances with their predefined settings.

The **pkicreate** script can be invoked after the packages are installed to create additional individual subsystem instances, with user-defined settings like the configuration and log directories and port numbers. After the instance is created, it is then configured through the HTML-based configuration wizard or by using the **pkisilent** script.

The syntax for **pkicreate** is slightly different between subsystems because of the different port configurations. The CA, OCSP, DRM, and TKS all have three SSL service ports and an unsecure service port, while the RA and TPS have two SSL service ports and an unsecure service port.

Example 5.1. Syntax for Creating a CA, OCSP, DRM, or TKS

```
pkicreate -pki_instance_root=/directory/path -subsystem_type=type -pki-
instance_name=instance_ID -secure_port=SSLport | {-agent_secure_port=SSLport -
ee_secure_port=SSLport -admin_secure_port=SSLport -
ee_secure_client_auth_port=SSLport_CA_only} -unsecure_port=port -
tomcat_server_port=port [-user=user_name] [-group=group_name] [-
redirect_conf=conf_directory] [-redirect_logs=log_directory]
```

Example 5.2. Syntax for Creating an RA or TPS

```
pkicreate -pki_instance_root=/directory/path -subsystem_type=type -pki-
instance_name=instance_ID -secure_port=SSLport -
non_clientauth_secure_port=SSLport -unsecure_port=port [-user=user_name] [-
group=group_name] [-redirect_conf=conf_directory] [-
redirect_logs=log_directory]
```

**TIP**

To get full usage examples and syntax for the **pkicreate** command, run **pkicreate --help**.

Table 5.1. pkicreate Parameters

| Parameter | Description |
|--------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>pki_instance_root</code> | Gives the full path to the new instance configuration directory. |
| <code>subsystem_type</code> | Gives the type of subsystem being created. |
| <code>pki_instance_name</code> | Gives the name of the new instance. Instance names must be unique on a single machine, but do not have to be unique within the security domain (since instances are identified by hostname and port, not instance name). |
| <code>secure_port</code> ^[a] | Sets a single SSL port number for the subsystem. This parameter is required if port separation is not configured, meaning that separate ports are not assigned for the administrator, agent, and end-entities services. |
| <code>agent_secure_port</code> ^[a] | Sets the SSL port for the agent web services. If this is specified, then both ee_secure_port and admin_secure_port must be specified. For CAs only, an end-entities client authentication port is also required with the ee_secure_client_auth_port option. |
| <code>ee_secure_port</code> ^[a] | Sets the SSL port for the end-entities web services. If this is specified, then both agent_secure_port and admin_secure_port must be specified. For CAs only, an end-entities client authentication port is also required with the ee_secure_client_auth_port option. |
| <code>ee_secure_client_auth_port</code> ^[a] | <i>For CAs only.</i> Sets the SSL port for the end-entity client authentication. If this is specified, then ee_secure_port , agent_secure_port , and admin_secure_port must be specified. |
| <code>admin_secure_port</code> ^[a] | Sets the SSL port number for the administrator services, usually the pkiconsole . If this is specified, then both agent_secure_port and ee_secure_port must be specified. For CAs only, an end-entities client authentication port is also required with the ee_secure_client_auth_port option. |
| <code>non_clientauth_secure_port</code> ^[a] | Sets the end entities SSL port for RA and TPS subsystems. |
| <code>unsecure_port</code> ^[a] | Sets the regular port number. If this is not set, the number is randomly generated. Still, it is recommended that administrators set this value to make sure there are no conflicts with SELinux labels for other services. |
| <code>tomcat_server_port</code> ^[a] | Sets the port number for the Tomcat web server for OSCP, TKS, and DRM instances. |
| <code>redirect_conf</code> | Sets the location for the configuration files for the |

| | |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|
| | new instance. |
| <code>redirect_logs</code> | Sets the location for the log files for the new instance. |
| <code>user</code> | Sets the user as which the Certificate System instance will run. This option must be set. |
| <code>group</code> | Sets the group as which the Certificate System instance will run. This option must be set. |
| [a] The ports selected for the new instance should not conflict with any other ports assigned on the host or SELinux. Check the <code>/etc/services</code> file to see port assignments for the system. Then, run <code>semanage port -l grep port#</code> to check SELinux; if there is no output, then there is no conflict with SELinux assignments. | |

For more information on the **pkicreate** tool options, see the *Certificate System Command-Line Tools Guide*.

5.2. Running pkicreate for a Single SSL Port

1. Run the **pkicreate** command, specifying the type of subsystem being created, the configuration directory, instance name, and port numbers. For example, this created a second DRM instance:

```
pkicreate -pki_instance_root=/var/lib -subsystem_type=kra -
pki_instance_name=pki-drm2 -secure_port=10543 -unsecure_port=10180 -
tomcat_server_port=1802 -verbose
```

2. When the instance is successfully created, the process returns a URL for the HTML configuration page. For example:

```
http://server.example.com:10180/kra/admin/console/config/login?
pin=nt2z2keqcqAZiBRBGldf
```



TIP

The configuration URL is written to the end of the instance's installation file, `/var/log/subsystem_name-install.log`. This log is also useful for debugging an instance.

3. Open the new instance URL, and go through the configuration wizard as described in [Chapter 3, Installation and Configuration](#). Supply the security domain, CA, instance ID, internal LDAP database, and agent information.
4. When the configuration is complete, restart the subsystem.

```
service instance_ID restart
```

5.3. Running pkicreate with Port Separation

To create an instance with three separate ports for the different subsystem services, run **pkicreate** with three options which specify the services ports: **-admin_secure_port**, **-agent_secure_port**, and **-ee_secure_port**. For CAs only, there is an additional port for end-entity client authentication, **-ee_secure_client_auth_port**.

Separated SSL ports is the default instance configuration because it is more secure than using a single

SSL port.

1. Run the **pkicreate** command. For example:

```
pkicreate -pki_instance_root=/var/lib -subsystem_type=ca -  
pki_instance_name=pki-ca2 -admin_secure_port=9545 -agent_secure_port=9544 -  
ee_secure_port=9543 -ee_secure_client_auth_port=9546 -unsecure_port=9180 -  
tomcat_server_port=1802 -verbose
```

2. When the instance is successfully created, the process returns a URL for the HTML configuration page. For example:

```
http://server.example.com:10180/kra/admin/console/config/login?  
pin=nt2z2keqcqAZiBRBGldf
```



TIP

The configuration URL is written to the end of the instance's installation file, **/var/log/*subsystem_name*-install.log**. This log is also useful for debugging an instance.

3. Open the new instance URL, and go through the configuration wizard as described in [Chapter 3, Installation and Configuration](#). Supply the security domain, CA, instance ID, internal LDAP database, and agent information.
4. When the configuration is complete, restart the subsystem.

```
service subsystem_name restart
```

For more information on the **pkicreate** tool options, see the *Certificate System Command-Line Tools Guide*.

Chapter 6. Cloning Subsystems

When a new subsystem instance is first configured, the Red Hat Certificate System allows subsystems to be cloned, or duplicated, for high availability of the Certificate System. The cloned instances run on different machines to avoid a single point of failure and their databases are synchronized through replication.

6.1. About Cloning

Planning for *high availability* reduces unplanned outages and other problems by making one or more subsystem clones available. When a host machine goes down, the cloned subsystems can handle requests and perform services, taking over from the master (original) subsystem seamlessly and keeping uninterrupted service.

Using cloned subsystems also allows systems to be taken offline for repair, troubleshooting, or other administrative tasks without interrupting the services of the overall PKI system.



NOTE

All of the subsystems except the TPS and RA can be cloned.

Cloning is one method of providing scalability to the PKI by assigning the same task, such as handling certificate requests, to separate instances on different machines. The internal databases for the master and its clones are replicated between each other, so the information about certificate requests or archived keys on one subsystem is available on all the others.

Typically, master and cloned instances are installed on different machines, and those machines are placed behind a *load balancer*. The load balancer accepts HTTP and HTTPS requests made to the Certificate System subsystems and directs those requests appropriately between the master and cloned instances. In the event that one machine fails, the load balancer transparently redirects all requests to the machine that is still running until the other machine is brought back online.

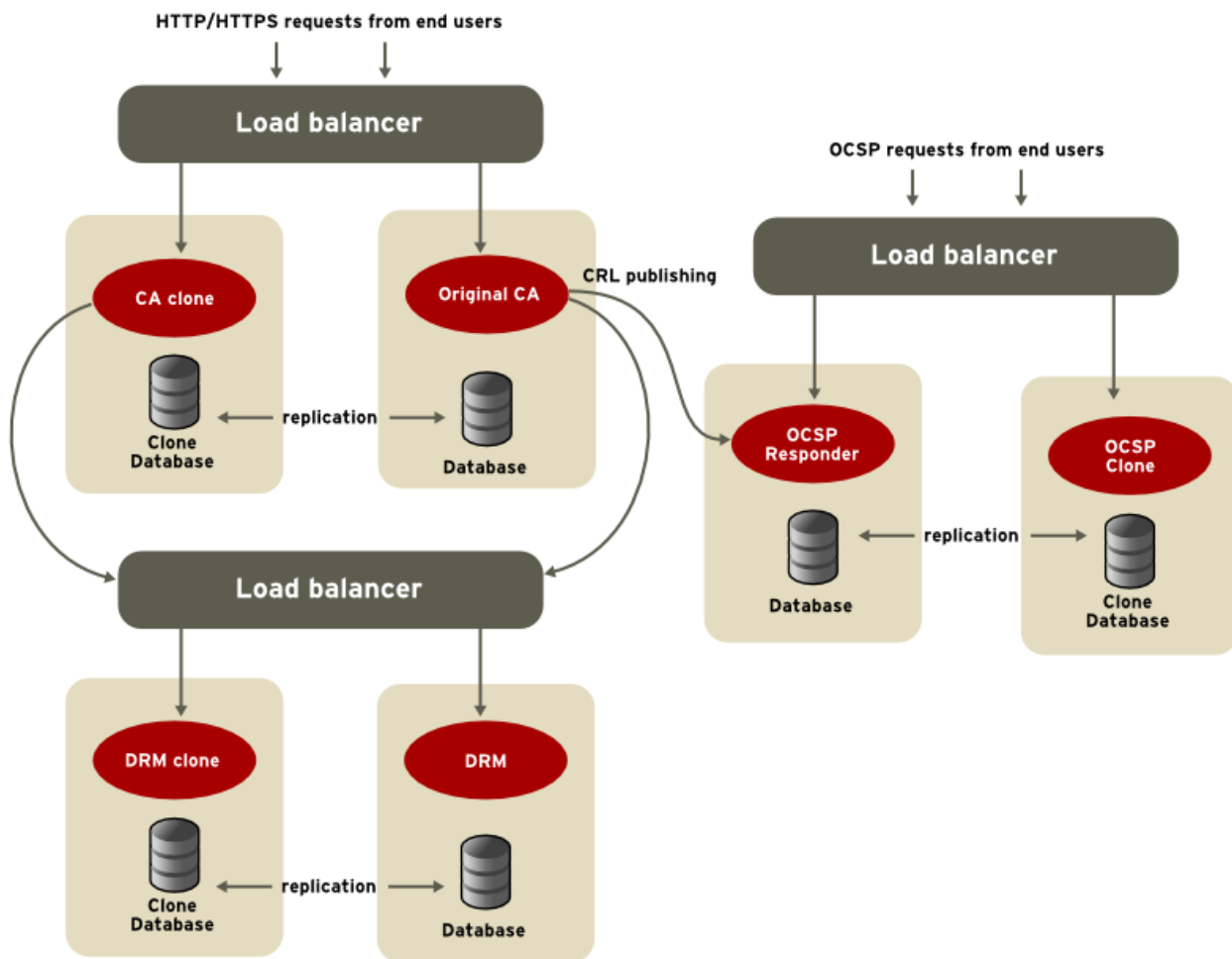


Figure 6.1. Cloning Example

The load balancer in front of a Certificate System subsystem is what provides the actual failover support in a high availability system. A load balancer can also provide the following advantages as part of a Certificate System subsystem:

- DNS round-robin, a feature for managing network congestion that distributes load across several different servers.
- Sticky SSL, which makes it possible for a user returning to the system to be routed the same host used previously.

6.1.1. Cloning for CAs

Cloned instances have the exact same private keys as the master, so their certificates are identical. For CAs, that means that the CA signing certificates are identical for the original master CA and its cloned CAs. From the perspectives of clients, these look like a single CA.

Every CA, both cloned and master, can issue certificates and process revocation requests.

The main issue with managing cloned CAs is how to assign serial numbers to the certificates they issue. Different CAs can have different levels of traffic, using serial numbers at different rates, and it is imperative that no CA issue the certificates with the same serial number. These serial number ranges are assigned and managed dynamically by using a shared, replicated entry that defines the ranges for

each CA and the next available range to reassign when one CA range runs low.

Cloned CAs do have limits on what operations they can perform. Most important, cloned CAs cannot generate or publish CRLs. Any CRL requests submitted to a cloned CA are immediately redirected to the master CA. Anything related to generating or caching CRLs is disabled in the **CS.cfg** file for the clone. Clones can revoke, display, import, and download CRLs previously generated by master CAs, but having them generate new CRLs may cause synchronization problems. Only a single CA should generate CRLs, and this task is always left to the master CA, which also maintains the CRL cache.

Master CAs also manage the relationships and information sharing between the cloned CAs by monitoring replication changes to the internal databases.

6.1.2. Cloning for DRMs

With DRMs, all keys archived in one DRM are replicated to the internal databases of the other DRMs. This allows a key recovery to be initiated on any clone DRM, regardless of which DRM the key was originally archived on.

After a key recovery is processed, the record of the recovery is stored in the internal database of all of the cloned DRMs.

Although key recovery can be initiated on any clone, once the recovery is initiated, it must be completed on the same single DRM. This is because a recovery operation is recorded in the replicated database only after the appropriate number of approvals have been obtained from the DRM agents. Until then, the DRM on which the recovery is initiated is the only one which knows about the recovery operation.

6.1.3. Cloning for Other Subsystems

There is no real operational difference between masters and clones for TKSs; the information created or maintained on one is replicated along the other servers.

For OCSPs, only the master OCSP receives CRL updates, and then the published CRLs are replicated to the clones.

6.1.4. Cloning and Key Stores

Cloning a subsystem creates two server processes performing the same functions: another new instance of the subsystem is created and configured to use the same keys and certificates to perform its operations. Depending on where the keys are stored for the master clone, the method for the clone to access the keys is very different.

If the keys and certificates are stored in the internal software token, then the keys must be exported from the master subsystem when it is first configured. When configuring the master instance, there is an option in the **Export Keys and Certificates** panel to back up the keys and certificates to a PKCS#12 file. Before the clone instance is configured, the PKCS#12 file is copied to the **alias/** directory for the clone instance. Then, the PKCS#12 filename is given in the **Restore Keys and Certificates** screen during the clone's configuration.

If the keys and certificates are stored on a hardware token, then the keys and certificates can be copied or referenced directly in the token:

- Duplicate all the required keys and certificates, except the SSL server key and certificate to the clone instance. Keep the nicknames for those certificates the same. Additionally, copy all the necessary trusted root certificates from the master instance to the clone instance, such as chains or cross-pair certificates.
- If the token is network-based, then the keys and certificates simply need to be available to the token;

the keys and certificates do not need to be copied.

- When using a network-based hardware token, make sure the high-availability feature is enabled on the hardware token to avoid single point of failure.

6.1.5. LDAP Considerations

As mentioned in [Section 6.1, “About Cloning”](#), part of the behavior of cloning is to replicate information between the master and the clone, so that they work from an identical set of data and records. This means that the LDAP servers for the master and clones need to be able to communicate. If these servers are on different hosts, then make sure that there is appropriate firewall access to allow the Directory Server instances to connect with each other.



NOTE

Cloned subsystems and their masters use separate LDAP servers.

6.2. Exporting Keys from a Software Database

Ideally, the keys for the master instance are exported when the instance is first created. If the keys were not exported then or if the backup file is lost, then it is possible to extract the keys from the internal software database for the subsystem instance using the **PKCS12Export** command. For example:

```
PKCS12Export -debug -d /var/lib/subsystem_name/alias -w p12pwd.txt -p internal.txt  
-o master.p12
```

The PKCS#12 file (in this example, **master.p12**) can then be copied to the clone instance's **alias/** directory and imported during the clone configuration.



NOTE

Keys and certificates do not need to be exported from an HSM, so long as the clone instance is installed using the same HSM as the master. If both instances use the same key store, then the keys are naturally available to the clone.

6.3. Cloning a CA

1. Configure the master CA, as described in [Section 3.3, “Configuring a CA”](#), and back up the keys.
2. In the **CS.cfg** file for the master CA, enable the master CA to monitor replication database changes by adding the **ca.listenToCloneModifications** parameter:

```
cd /etc/subsystem_name  
  
ca.listenToCloneModifications=true
```

3. Create the clone subsystem instance.



IMPORTANT

Do *not* go through the setup wizard for the instance yet.

4. Copy the exported PKCS#12 file containing the master instance's keys to the clone's **alias/** directory.

The keys for the master instance could have been exported to a **.p12** file when the instance was configured. Alternatively, the keys can be exported using the **PKCS12Export** command, as in [Section 6.2, “Exporting Keys from a Software Database”](#).

5. Make sure the PKCS#12 file is accessible by the Certificate System user. If necessary, change the file owner to **pkiuser** and reset the permissions to allow the correct read/write access. For example:

```
chown pkiuser:pkiuser example.p12
chmod 00644 example.p12
```

6. It may be necessary to reset the SELinux permissions for the exported file so that the setup program can use it. First, check what context is assigned:

```
ls -lZ *
-rw----- . pkiuser pkiuser system_u:object_r:pki_ca_var_lib_t:s0 cert8.db
-rw----- . pkiuser pkiuser system_u:object_r:pki_ca_var_lib_t:s0 key3.db
-rw-r--r-- . pkiuser pkiuser system_u:object_r:nfs_t:s0 example.p12
-rw----- . pkiuser pkiuser system_u:object_r:pki_ca_var_lib_t:s0 secmod.db
```

If it does not match the other security files, then reset the SELinux context to that of the other objects using **chcon**:

```
chcon "system_u:object_r:pki_ca_var_lib_t:s0" example.p12

ls -lZ *
-rw----- . pkiuser pkiuser system_u:object_r:pki_ca_var_lib_t:s0 cert8.db
-rw----- . pkiuser pkiuser system_u:object_r:pki_ca_var_lib_t:s0 key3.db
-rw-r--r-- . pkiuser pkiuser system_u:object_r:pki_ca_var_lib_t:s0
example.p12
-rw----- . pkiuser pkiuser system_u:object_r:pki_ca_var_lib_t:s0 secmod.db
```

7. Open the setup wizard URL, which was returned when the instance was created. For example:

```
http://server.example.com:9180/ca/admin/console/config/login?
pin=HIsd90RJSioDK==
```

8. In the **Security Domain** panel, add the clone to the same security domain to which the master belongs.
9. The **Subsystem Type** panel sets whether to create a new instance or a clone; select the clone radio button.

Welcome
 Security Domain
 Subsystem Type
 PKI Hierarchy
 Internal Database
 Key Store
 Key Pairs
 Subject Names
 Requests and
 Certificates
 Export Keys and
 Certificates
 Import CA's
 Certificate Chain
 Administrator
 Done

Subsystem Type

Subsystem Configuration

This instance can be configured as a new CA subsystem.

☐ Configure this Instance as a New CA Subsystem

Subsystem Name: (e.g. - Certificate Authority)

Subsystem HTTP EE URL (unsecure):

Subsystem HTTPS Agent URL (clientauth):

Subsystem HTTPS EE URL (non-clientauth):

Subsystem HTTPS Admin URL (non-clientauth):

☐ Clone an Existing CA Subsystem

Subsystem Name: (e.g. - Certificate Authority Clone 1)

Subsystem URL:

Next>

10. Give the path and filename of the PKCS #12 backup file which was saved when the master instance was created or that were exported in 4.

If the keys are stored on an HSM that is accessible to the clone, then they are picked up automatically.

Welcome
 Security Domain
 Subsystem Type
 PKI Hierarchy
 Internal Database
 Security Modules
 Key Pairs
 Subject Names
 Requests and
 Certificates
 Export Keys and
 Certificates
 Administrator

Restore Keys and Certificates

Restore Keys and Certificates

For cloning setup, the subsystem keys and certificates (except for the SSL server certificate) need to be restored. They are in the pk12 format.

Path where the pk12 files are located:

PK12 Password:

<Back **Next>**



NOTE

When cloning a CA, the master and clone instances have the same CA signing key.

11. The subsystem information is automatically supplied from the master instance to the clone instance once the keys are successfully restored. Complete the configuration process.



NOTE

By default, the instance configuration wizard uses **localhost** as the location for the internal LDAP database for a new instance. However, with cloning, the configuration process will spin endlessly and never complete if localhost is used for the internal database location, even if the LDAP database is indeed installed on the localhost. Use the fully-qualified domain name for the LDAP database in the **Internal Database** panel when configuring a clone.

12. Edit the **CS.cfg** file for the clone. Certain parameters must be added to the clone configuration to disable caching and generating CRLs.

- Disable control of the database maintenance thread:

```
ca.certStatusUpdateInterval=0
```

- Disable monitoring database replication changes:

```
ca.listenToCloneModifications=false
```

- Disable maintenance of the CRL cache:

```
ca.crl.IssuingPointId.enableCRLCache=false
```

- Disable CRL generation:

```
ca.crl.IssuingPointId.enableCRLUpdates=false
```

- Enable the clone to redirect CRL requests to the master clone:

```
master.ca.agent.host=master_hostname
master.ca.agent.port=master_port
```

13. Restart the clone instance.

```
service subsystem_name restart
```

After configuring the clone, test to make sure that the master-clone relationship is functioning:

1. Request a certificate from the cloned CA.
2. Approve the request.
3. Download the certificate to the browser.
4. Revoke the certificate.
5. Check master CA's CRL for the revoked certificate. In the master Certificate Manager's agent services page, click **Update Certificate Revocation List**. Find the CRL in the list. The CRL should show the certificate revoked by the cloned Certificate Manager. If that certificate is not listed, check logs to resolve the problem.

6.4. Cloning OCSP Subsystems

1. Configure the master OCSP, as described in [Section 3.5, “Configuring a DRM, OCSP, or TKS”](#), and back up the keys.
2. In the **CS.cfg** file for the master OCSP, set the **OCSP.Responder.store.defStore.refreshInSec** parameter to any non-zero number other than 21600; 21600 is the setting for a clone.

```
vim /etc/subsystem_name/CS.cfg

OCSP.Responder.store.defStore.refreshInSec=15000
```

3. Create the clone subsystem instance.



IMPORTANT

Do not go through the setup wizard for the instance yet.

4. Copy the exported PKCS#12 file containing the master instance's keys to the clone's **alias/** directory.
The keys for the master instance could have been exported to a **.p12** file when the instance was configured. Alternatively, the keys can be exported using the **PKCS12Export** command, as in [Section 6.2, “Exporting Keys from a Software Database”](#).
5. Make sure the PKCS#12 file is accessible by the Certificate System user. If necessary, change the file owner to **pkiuser** and reset the permissions to allow the correct read/write access. For example:

```
chown pkiuser:pkiuser example.p12
chmod 00644 example.p12
```

6. It may be necessary to reset the SELinux permissions for the exported file so that the setup program can use it. First, check what context is assigned:

```
ls -lZ *
-rw----- . pkiuser pkiuser system_u:object_r:pki_ocsp_var_lib_t:s0 cert8.db
-rw----- . pkiuser pkiuser system_u:object_r:pki_ocsp_var_lib_t:s0 key3.db
-rw-r--r-- . pkiuser pkiuser system_u:object_r:nfs_t:s0 example.p12
-rw----- . pkiuser pkiuser system_u:object_r:pki_ocsp_var_lib_t:s0 secmod.db
```

If it does not match the other security files, then reset the SELinux context to that of the other objects using **chcon**:

```
chcon "system_u:object_r:pki_ocsp_var_lib_t:s0" example.p12

ls -lZ *
-rw----- . pkiuser pkiuser system_u:object_r:pki_ocsp_var_lib_t:s0 cert8.db
-rw----- . pkiuser pkiuser system_u:object_r:pki_ocsp_var_lib_t:s0 key3.db
-rw-r--r-- . pkiuser pkiuser system_u:object_r:pki_ocsp_var_lib_t:s0
example.p12
-rw----- . pkiuser pkiuser system_u:object_r:pki_ocsp_var_lib_t:s0 secmod.db
```

7. Open the setup wizard URL, which was returned when the instance was created. For example:

```
http://server.example.com:11180/ocsp/admin/console/config/login?
pin=I0jh7fI0jklD90k
```

8. In the **Security Domain** panel, add the clone to the same security domain to which the master belongs.
9. The **Subsystem Type** panel sets whether to create a new instance or a clone; select the clone radio button.

Welcome
Security Domain
Subsystem Type
PKI Hierarchy
Internal Database
Key Store
Key Pairs
Subject Names
Requests and Certificates
Export Keys and Certificates
Import CA's Certificate Chain
Administrator
Done

Subsystem Type

Subsystem Configuration

This instance can be configured as a new CA subsystem.

☐ **Configure this Instance as a New CA Subsystem**

Subsystem Name: (e.g. - Certificate Authority)

Subsystem HTTP EE URL (unsecure):

Subsystem HTTPS Agent URL (clientauth):

Subsystem HTTPS EE URL (non-clientauth):

Subsystem HTTPS Admin URL (non-clientauth):

☒ **Clone an Existing CA Subsystem**

Subsystem Name: (e.g. - Certificate Authority Clone 1)

Subsystem URL: ▼

Next>

10. Give the path and filename of the PKCS #12 backup file which was saved when the master instance was created or that were exported in [3](#).
 If the keys are stored on an HSM that is accessible to the clone, then they are picked up automatically.

Welcome
Security Domain
Subsystem Type
PKI Hierarchy
Internal Database
Security Modules
Key Pairs
Subject Names
Requests and Certificates
Export Keys and Certificates
Administrator

Restore Keys and Certificates

Restore Keys and Certificates

For cloning setup, the subsystem keys and certificates (except for the SSL server certificate) need to be restored. They are in the pk12 format.

Path where the pk12 files are located:

PK12 Password:

<Back **Next>**

11. The subsystem information is automatically supplied from the master instance to the clone instance once the keys are successfully restored. Complete the configuration process.

**NOTE**

By default, the instance configuration wizard uses **localhost** as the location for the internal LDAP database for a new instance. However, with cloning, the configuration process will spin endlessly and never complete if localhost is used for the internal database location, even if the LDAP database is indeed installed on the localhost. Use the fully-qualified domain name for the LDAP database in the **Internal Database** panel when configuring a clone.

12. Edit the **CS.cfg** file for the clone. Set the **OCSP.Responder.store.defStore.refreshInSec** parameter in the clone instance to **21600**.

```
vim /etc/subsystem_name/CS.cfg

OCSP.Responder.store.defStore.refreshInSec=21600
```

13. Restart the clone instance.

```
service subsystem_name start
```

After configuring the clone, test to make sure that the master-clone relationship is functioning:

1. Set up OCSP publishing in the master CA so that the CRL is published to the master OCSP.
2. Once the CRL is successfully published, check both the master and cloned OCSP's **List Certificate Authorities** link in the agent pages. The list should be identical.
3. Use the **OCSPClient** tool to submit OCSP requests to the master and the cloned Online Certificate Status Manager. The tool should receive identical OCSP responses from both managers.

6.5. Cloning DRM and TKS Subsystems

1. Configure the master subsystem, as described in [Section 3.5, “Configuring a DRM, OCSP, or TKS”](#), and back up the keys.
2. Create the clone subsystem instance.

**IMPORTANT**

Do *not* go through the setup wizard for the instance yet.

3. Copy the exported PKCS#12 file containing the master instance's keys to the clone's **alias/** directory.
The keys for the master instance could have been exported to a **.p12** file when the instance was configured. Alternatively, the keys can be exported using the **PKCS12Export** command, as in [Section 6.2, “Exporting Keys from a Software Database”](#).
4. Make sure the PKCS#12 file is accessible by the Certificate System user. If necessary, change the file owner to **pkiuser** and reset the permissions to allow the correct read/write access. For example:

```
chown pkiuser:pkiuser example.p12
chmod 00644 example.p12
```

5. It may be necessary to reset the SELinux permissions for the exported file so that the setup program can use it. First, check what context is assigned:

```
ls -lZ *
-rw----- . pkiuser pkiuser system_u:object_r:pki_tks_var_lib_t:s0 cert8.db
-rw----- . pkiuser pkiuser system_u:object_r:pki_tks_var_lib_t:s0 key3.db
-rw-r--r-- . pkiuser pkiuser system_u:object_r:nfs_t:s0 example.p12
-rw----- . pkiuser pkiuser system_u:object_r:pki_tks_var_lib_t:s0 secmod.db
```

If it does not match the other security files, then reset the SELinux context to that of the other objects using **chcon**:

```
chcon "system_u:object_r:pki_tks_var_lib_t:s0" example.p12

ls -lZ *
-rw----- . pkiuser pkiuser system_u:object_r:pki_tks_var_lib_t:s0 cert8.db
-rw----- . pkiuser pkiuser system_u:object_r:pki_tks_var_lib_t:s0 key3.db
-rw-r--r-- . pkiuser pkiuser system_u:object_r:pki_tks_var_lib_t:s0
example.p12
-rw----- . pkiuser pkiuser system_u:object_r:pki_tks_var_lib_t:s0 secmod.db
```

6. Open the setup wizard URL, which was returned when the instance was created. For example:

```
http://server.example.com:10180/kra/admin/console/config/login?
pin=Jhu7HBJiodljnw3oijs
```

7. In the **Security Domain** panel, add the clone to the same security domain to which the master belongs.
8. The **Subsystem Type** panel sets whether to create a new instance or a clone; select the clone radio button.

Welcome
 Security Domain
Subsystem Type
 PKI Hierarchy
 Internal Database
 Key Store
 Key Pairs
 Subject Names
 Requests and
 Certificates
 Export Keys and
 Certificates
 Import CA's
 Certificate Chain
 Administrator
 Done

Subsystem Type

Subsystem Configuration

This instance can be configured as a new CA subsystem.

☐ **Configure this Instance as a New CA Subsystem**

Subsystem Name: (e.g. - Certificate Authority)

Subsystem HTTP EE URL (unsecure):

Subsystem HTTPS Agent URL (clientauth):

Subsystem HTTPS EE URL (non-clientauth):

Subsystem HTTPS Admin URL (non-clientauth):

☐ **Clone an Existing CA Subsystem**

Subsystem Name: (e.g. - Certificate Authority Clone 1)

Subsystem URL:

9. Give the path and filename of the PKCS #12 backup file which was saved when the master instance was created or that were exported in [3.](#)

If the keys are stored on an HSM that is accessible to the clone, then they are picked up automatically.

Welcome
 Security Domain
 Subsystem Type
 PKI Hierarchy
 Internal Database
 Security Modules
 Key Pairs
 Subject Names
 Requests and
 Certificates
 Export Keys and
 Certificates
Administrator

Restore Keys and Certificates

Restore Keys and Certificates

For cloning setup, the subsystem keys and certificates (except for the SSL server certificate) need to be restored. They are in the pk12 format.

Path where the pk12 files are located:

PK12 Password:



NOTE

When cloning a DRM, the master and clone instances have the same storage and transport keys.

10. The subsystem information is automatically supplied from the master instance to the clone instance once the keys are successfully restored. Complete the configuration process.

**NOTE**

By default, the instance configuration wizard uses **localhost** as the location for the internal LDAP database for a new instance. However, with cloning, the configuration process will spin endlessly and never complete if localhost is used for the internal database location, even if the LDAP database is indeed installed on the localhost. Use the fully-qualified domain name for the LDAP database in the **Internal Database** panel when configuring a clone.

11. Restart the clone instance.

```
service subsystem_name restart
```

For the DRM clone, test to make sure that the master-clone relationship is functioning:

1. Go to the DRM agent's page.
2. Click **List Requests**.
3. Select **Show all requests** for the request type and status.
4. Click **Submit**.
5. Compare the results from the cloned DRM and the master DRM. The results ought to be identical.

For the TKS, enroll a smart card and then run an **ldapsearch** to make sure that the same key information is contained in both databases.

6.6. Converting Masters and Clones

There can be any number of clones, but there can only be a single configured master. For DRMs and TKSs, there is no configuration difference between masters and clones, but CAs and OCSPs do have some configuration differences. This means that when a master is taken offline — because of a failure or for maintenance or to change the function of the subsystem in the PKI — then the existing master must be reconfigured to be a clone, and one of the clones promoted to be the master.

6.6.1. Converting CA Clones and Masters

1. Stop the master CA if it is still running.
2. Open the existing master CA configuration directory:

```
cd /var/lib/pki-ca/conf
```

3. Edit the **CS.cfg** file for the master, and change the CRL and maintenance thread settings so that it is set as a clone:

- Disable control of the database maintenance thread:

```
ca.certStatusUpdateInterval=0
```

- Disable monitoring database replication changes:

```
ca.listenToCloneModifications=false
```

- Disable maintenance of the CRL cache:

```
ca.cr1.IssuingPointId.enableCRLCache=false
```

- Disable CRL generation:

```
ca.cr1.IssuingPointId.enableCRLUpdates=false
```

- Set the CA to redirect CRL requests to the new master:

```
master.ca.agent.host=new_master_hostname  
master.ca.agent.port=new_master_port
```

4. Stop the cloned CA server.

```
service subsystem_name stop
```

5. Open the cloned CA's configuration directory.

```
cd /etc/subsystem_name
```

6. Edit the **CS.cfg** file to configure the clone as the new master.

- a. Delete each line which begins with the **ca.cr1.** prefix.
- b. Copy each line beginning with the **ca.cr1.** prefix from the former master CA **CS.cfg** file into the cloned CA's **CS.cfg** file.
- c. Enable control of the database maintenance thread; the default value for a master CA is **600**.

```
ca.certStatusUpdateInterval=600
```

- d. Enable monitoring database replication:

```
ca.listenToCloneModifications=true
```

- e. Enable maintenance of the CRL cache:

```
ca.cr1.IssuingPointId.enableCRLCache=true
```

- f. Enable CRL generation:

```
ca.cr1.IssuingPointId.enableCRLUpdates=true
```

- g. Disable the redirect settings for CRL generation requests:

```
master.ca.agent.host=hostname  
master.ca.agent.port=port number
```

7. Start the new master CA server.

```
service subsystem_name start
```

6.6.2. Converting OCSP Clones

1. Stop the OCSP master, if it is still running.
2. Open the existing master OCSP configuration directory.

```
cd /etc/subsystem_name
```

3. Edit the **CS.cfg**, and reset the **OCSP.Responder.store.defStore.refreshInSec** parameter to **21600**:

```
OCSP.Responder.store.defStore.refreshInSec=21600
```

4. Stop the online cloned OCSP server.

```
service subsystem_name stop
```

5. Open the cloned OCSP responder's configuration directory.

```
cd /etc/subsystem_name
```

6. Open the **CS.cfg** file, and delete the **OCSP.Responder.store.defStore.refreshInSec** parameter or change its value to any non-zero number:

```
OCSP.Responder.store.defStore.refreshInSec=15000
```

7. Start the new master OCSP responder server.

```
service subsystem_name start
```

6.7. Updating CA Clones

When a CA is cloned, any configuration in its **CS.cfg** is also copied to the clone CA. This includes any DRMs which are configured for the CA to use for key archival. However, if a DRM is configured for a master CA *after* a clone is created, then the new DRM configuration must be copied over to the clone CAs manually.

To update any clone CAs with new DRM configuration in the master CA:

1. Stop the clone CA.

```
service subsystem_name stop
```

Always stop a subsystem instance before editing its configuration files.

2. Copy all of the **ca.connector.KRA.*** parameters for the new DRM connection in the **CS.cfg** for the master CA over to the clone CA **CS.cfg** file.
3. Restart the clone CA.

```
service subsystem_name restart
```

Chapter 7. Silent Configuration

The Certificate System includes a tool, **pkisilent**, which configures an instance in a single step. Normally, instances are configured by accessing the subsystem HTML page and going through the setup wizard. **pkisilent** can be used to pass all of the configuration parameters to a new instance simply from the command line.



NOTE

The **pkisilent** script is downloaded and installed in its own package.

7.1. About pkisilent

Silent configuration sets up a new subsystem instance in a single pass, by sending all of the configuration parameters through the command line. For Certificate System subsystems, this is done using the **pkisilent** command.

The **pkisilent** command can configure the subsystem instance the same as if it were configured using the HTML-based configuration wizard, so it can create a new security domain or use an existing one, back up keys, create a clone, or use certificates issued by an external CA.

From a high level, the **pkisilent** command has groups of parameters that define major areas of the subsystem's default settings and users.

There are two template files that are shell scripts for silent configuration:

/usr/share/pki/silent/pki_silent.template and

/usr/share/pki/silent/subca_silent.template. Both of these templates have detailed information on parameters and usage options for **pkisilent**.

Example 7.1. pkisilent Command

```
pkisilent Configuretype -parameters to configure the subsystem URL ... -
parameters to configure the admin user ... -parameters to configure the domain
... -parameters to configure the agent ... -parameters to configure the
internal database ... -parameters to configure the subsystem keys,
certificates, and key store
```

The options available to use with the **pkisilent** command are listed in [Table 7.1, “Parameters for pkisilent”](#).

**TIP**

There are two template files that are shell scripts for silent configuration:

/usr/share/pki/silent/pki_silent.template and

/usr/share/pki/silent/subca_silent.template. Both of these templates have detailed information on parameters and usage options for **pkisilent**.

To check the specific options for any **Configuretype** option, just run the **pkisilent** command with the **Configuretype** option and the **-help** flag. For example, to get the help for configuring a subordinate CA:

```
pkisilent ConfigureSubCA -help
```

The **Configuretype** option sets what kind of subsystem is being configured. This can be any of the following:

- ConfigureCA (for a root CA) or ConfigureSubCA (for a subordinate CA)
- ConfigureRA
- ConfigureDRM
- ConfigureOCSP
- ConfigureTKS
- ConfigureTPS

Table 7.1. Parameters for pkisilent

| Parameter | Description |
|-------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Basic Instance Configuration | |
| cs_hostname | The hostname for the Certificate System machine. |
| cs_port | The administrative SSL port number of the Certificate System instance. |
| subsystem_name | Sets the name of the new subsystem instance. |
| client_certdb_dir | The directory for the subsystem certificate databases. |
| client_certdb_pwd | The password to protect the certificate database. |
| preop_pin | The preoperation PIN number used for the initial configuration. This PIN is part of the output of pkicreate , at the end of the configuration URL. It can also be found in the URL in the installation file for the instance (/var/log/subsystem_name-install.log). |
| token_name | Gives the name of the HSM token used to store the subsystem certificates. This is only required for hardware tokens; if this parameter is not given, then the script automatically uses the local software token. |
| token_pwd | Gives the password for the HSM. |
| Agent and Admin User Configuration | |
| admin_user | The new admin user for the new subsystem. |
| admin_email | The email address of the admin user. |
| admin_password | The password for the admin user. |
| agent_key_size | The key size to use for generating the agent certificate and key pair. |
| agent_key_type | The key type to use for generating the agent certificate and key pair. |
| agent_cert_subject | The subject name for the agent certificate. |
| Security Domain Configuration | |
| domain_name | The name of the security domain to which the subsystem will be added. |
| sd_hostname | The hostname of the CA which hosts security domain. |
| sd_admin_port | The administrative SSL port of the CA which hosts security domain. |
| sd_agent_port | The agent SSL port of the CA which hosts security domain. |
| sd_ssl_port | The end-entities SSL port of the CA which hosts security domain. |
| sd_admin_name | The username of the administrative user for the CA hosting the security domain. |
| sd_admin_password | The password of the administrative user for the CA hosting the security domain. |
| Internal Database Configuration | |

| | |
|------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ldap_host | The hostname of the Directory Server machine. |
| ldap_port | The non-SSL port of the Directory Server. |
| bind_dn | The bind DN which will access the Directory Server; this is normally the Directory Manager ID. |
| bind_password | The bind DN password. |
| base_dn | The entry DN under which to create all of the subsystem entries. |
| db_name | The database name. |
| Subsystem Certificates and Keys Configuration | |
| key_size | The size of the key to generate. The recommended size for an RSA key is 1048 bits for regular operations and 2048 bits for sensitive operations. |
| key_type | The type of key to generate; the only option is RSA. |
| key_algorithm | <p>The hashing algorithm to use for the key pair. This is only used for root CA subsystems; hashing algorithms for other subsystems and sub CAs are set by editing the certificate profile. For RSA:</p> <ul style="list-style-type: none"> ▸ SHA256withRSA ▸ SHA1withRSA ▸ SHA256withRSA ▸ SHA512withRSA ▸ MD5withRSA ▸ MD2withRSA |
| save_p12 | Sets whether to export the keys and certificate information to a backup PKCS #12 file. true backs up the information; false does not back up the information. <i>Only for the CA subsystem.</i> |
| backup_pwd | The password to protect the PKCS #12 backup file containing the subsystem keys and certificates. <i>Not for use with TPS installation.</i> |
| backup_fname | The file to which to export the the PKCS #12 backup file. |
| ca_subsystem_cert_subject_name | The subject names for the CA subsystem certificates. |
| ca_ocsp_cert_subject_name | |
| ca_server_cert_subject_name | |
| ca_sign_cert_subject_name | |
| ca_audit_signing_cert_subject_name | |
| ra_subsystem_cert_subject_name | The subject names and nicknames for the RA subsystem certificates. |
| ra server cert subject name | |

| | |
|--------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| ra_subsystem_cert_nickname | |
| ra_server_cert_nickname | |
| ocsp_ocsp_cert_subject_name | The subject names for the OCSP subsystem certificates. |
| ocsp_server_cert_subject_name | |
| ocsp_subsystem_cert_subject_name | |
| ocsp_audit_signing_cert_subject_name | |
| drm_storage_cert_subject_name | The subject names for the DRM subsystem certificates. |
| drm_transport_cert_subject_name | |
| drm_server_cert_subject_name | |
| drm_subsystem_cert_subject_name | |
| drm_audit_signing_cert_subject_name | |
| tkc_subsystem_cert_subject_name | The subject names for the TKS subsystem certificates. |
| tkc_server_cert_subject_name | |
| tkc_audit_signing_cert_subject_name | |
| tps_subsystem_cert_subject_name | The subject names and nicknames for the TPS subsystem certificates. |
| tps_server_cert_subject_name | |
| tps_subsystem_cert_nickname | |
| tps_server_cert_nickname | |
| Required Subsystem Configuration | |
| ca_hostname | The hostname for the CA subsystem which will issue the certificates for a subordinate CA, RA, DRM, OCSP, TKS, or TPS subsystem. |
| ca_port | The non-SSL port number of the CA. |
| ca_ssl_port | The SSL end entities port number of the CA. |
| drm_hostname | The hostname for the DRM subsystem to use to archive keys. <i>For the TPS only.</i> |
| drm_ssl_port | The SSL agent port number of the DRM. <i>For the TPS only.</i> |
| tkc_hostname | The hostname for the TKS subsystem to use to derive keys. <i>For the TPS only.</i> |
| tkc_ssl_port | The SSL agent port number of the TKS. <i>For the TPS only.</i> |
| Authentication Database Configuration (TPS only) | |

| | |
|---------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ldap_auth_host | Gives the hostname of the LDAP directory database to use for the TPS subsystem token database. <i>Only for the TPS subsystem.</i> |
| ldap_auth_port | Gives the port number of the LDAP directory database to use for the TPS subsystem token database. <i>Only for the TPS subsystem.</i> |
| ldap_auth_base_dn | Gives the base DN in the LDAP directory tree of the TPS token database under which to create token entries. <i>Only for the TPS subsystem.</i> |
| External CA for Issuing Certificates | |
| external | Sets whether to submit the subsystem certificates to the configured CA or to an external CA. The options are true or false . If this is not set, then the default is false . |
| ext_csr_file | The output file to which to write the generated certificate requests for the subsystem certificates. <i>Step one of the silent configuration process.</i> |
| ext_ca_cert_file | The input file for the certificates issued by the external CA. <i>Step two of the silent configuration process.</i> |
| ext_ca_cert_chain_file | The input file for the CA certificate chain for the external CA issuing the certificate. <i>Step two of the silent configuration process.</i> |
| Cloning Configuration | |
| clone | Sets whether the new instance is a clone. Its possible values are true or false . If this is not set, then the default is false . |
| clone_p12_file | The path and file name of the PKCS#12 file for the backed-up keys for the original instance. |
| clone_p12_password | The password to access the PKCS#12 file. |

7.2. Silently Configuring Subsystem



NOTE

Before running **pkisilent**, first run **pkicreate** to create the instance.

There are slight differences in the options used to configure the different subsystem types:

- Different security domain settings. A CA can host a security domain, so it has special configuration options to create a security domain. All other subsystems (as well as CAs) must join an existing security domain.

**TIP**

It is recommended that every CA have its own security domain, because each system within the security domain depends on having the security domain running and accessible. However, subordinate CAs can only be configured within the root CA's security domain using the **pkisilent** script.

- Different numbers and types of SSL ports. The CA, DRM, OCSP, and TKS each have three SSL ports (admin, agents, and users), while the RA and TPS both have two SSL ports (client and non-client).
- Different numbers and types of certificates.
- Different required subsystems. Every subsystem must, at a minimum, specify which CA will sign and issue its certificates, while a CA has the option of self-signing its certificates. The TPS also relies on a TKS and optional DRM, which can also be specified at configuration.
- Different database configuration. The RA uses a SQLite database as its internal databases, while all other subsystems use an LDAP directory. The TPS uses two separate LDAP directories, one as its internal database and the other as an authentication directory to help manage its users.

For all of that, the usage of **pkisilent** is still pretty similar between the subsystems. They use the same options to identify the instance to configure, back up their keys, and configure their users, and even though the parameters are slightly different in name, the configuration concepts (like cloning or generating certificates) are the same.

**NOTE**

Any spaces in the arguments used with **pkisilent** must be escaped.

[Example 7.2, “Configuring a Root CA”](#) configures a CA, creates a new security domain, backs up its keys, and self-signs its certificates.

Example 7.2. Configuring a Root CA

```
pkisilent ConfigureCA -cs_hostname localhost -cs_port 9445 -subsystem_name "pki-
ca2" -client_certdb_dir /tmp/ -client_certdb_pwd password -preop_pin
sYY8er834FG9793fsef7et5 -domain_name "testca" -admin_user admin -admin_email
"admin@example.com" -admin_password secret -agent_key_size 2048 -agent_key_type
rsa -agent_cert_subject "cn=ca\ agent\ cert" -ldap_host server -ldap_port 389 -
bind_dn "cn=directory\ manager" -bind_password secret -base_dn "o=pki-ca2" -
db_name "server.example.com-pki-ca2" -key_size 2048 -key_type rsa -
key_algorithm SHA256withRSA -save_p12 true -backup_pwd password -backup_fname
/export/backup.p12 -ca_subsystem_cert_subject_name "cn=ca\ subsystem\
cert,o=testca\ domain" -ca_ocsp_cert_subject_name "cn=ocsp\ signing\
cert,o=testca\ domain" -ca_server_cert_subject_name "cn=ca\ client\
cert,o=testca\ domain" -ca_sign_cert_subject_name "cn=ca\ signing\
cert,o=testca\ domain" -ca_audit_signing_cert_subject_name "cn=audit\ signing\
cert,o=testca\ domain"
```

A subordinate CA — along with the DRM, OCSP, and TKS — is configured to join an existing security domain and to have its certificates signed by an existing Certificate System CA (by default; it is also possible to use an external CA, as in [Section 7.4, “Performing Silent Configuration Using an External CA”](#)).

Example 7.3. Configuring a Subordinate CA

```
pkisilent ConfigureCA -cs_hostname localhost -cs_port 9445 -subsystem_name "pki-
ca2" -client_certrdb_dir /tmp/ -client_certrdb_pwd password -preop_pin
sYY8er834FG9793fsef7et5 -sd_hostname "domain.example.com" -sd_admin_port 9445
-sd_agent_port 9443 -sd_ssl_port 9444 -sd_admin_name admin -
sd_admin_password secret -admin_user admin -admin_email "admin@example.com" -
admin_password secret -agent_key_size 2048 -agent_key_type rsa -
agent_cert_subject "cn=ca\ agent\ cert" -ldap_host server -ldap_port 389 -
bind_dn "cn=directory\ manager" -bind_password secret -base_dn "o=pki-ca2" -
db_name "server.example.com-pki-ca2" -key_size 2048 -key_type rsa -save_p12
true -backup_pwd password -backup_fname /export/backup.p12 -ca_hostname
server.example.com ca_port 9180 -ca_ssl_port 9443 -
ca_subsystem_cert_subject_name "cn=ca\ subsystem\ cert,o=testca\ domain" -
ca_ocsp_cert_subject_name "cn=ocsp\ signing\ cert,o=testca\ domain" -
ca_server_cert_subject_name "cn=ca\ client\ cert,o=testca\ domain" -
ca_sign_cert_subject_name "cn=ca\ signing\ cert,o=testca\ domain" -
ca_audit_signing_cert_subject_name "cn=audit\ signing\ cert,o=testca\ domain"
```

The RA, unlike the other subsystems, does not use an LDAP database, so it does not specify the same database parameters as the other subsystems. In this example, the keys for the RA are not automatically backed up and there is no audit log signing certificate, since the RA is the only subsystem which does not support signed audit logs.

Example 7.4. Configuring an RA

```
pkisilent ConfigureRA -cs_hostname localhost -cs_port 9445 -subsystem_name "pki-
ra2" -client_certrdb_dir /tmp/ -client_certrdb_pwd password -preop_pin
sYY8er834FG9793fsef7et5 -sd_hostname "domain.example.com" -sd_admin_port 9445 -
sd_agent_port 9443 -sd_ssl_port 9444 -sd_admin_name admin -sd_admin_password
secret -admin_user admin -admin_email "admin@example.com" -admin_password
secret -agent_key_size 2048 -agent_key_type rsa -agent_cert_subject "cn=ra\
agent\ cert" -ca_hostname server.example.com -ca_port 9180 -ca_ssl_port 9443 -
key_size 2048 -key_type rsa -ra_subsystem_cert_subject_name "cn=ra\ subsystem\
cert,o=testca\ domain" -ra_server_cert_subject_name "cn=ra\ client\
cert,o=testca\ domain"
```

A TPS requires the most parameters, since it depends on having a CA, DRM, and TKS configured and uses two LDAP databases, along with joining an existing security domain. However, since the TPS cannot be cloned, it is not required to back up its keys to a PKCS #12 file.

Example 7.5. Configuring a TPS

```
pkisilent ConfigureTPS -cs_hostname localhost -cs_port 9445 -subsystem_name
"pki-tps2" -client_certdb_dir /tmp/ -client_certdb_pwd password -preop_pin
sYY8er834FG9793fsef7et5 -sd_hostname "domain.example.com" -sd_admin_port 9445 -
sd_agent_port 9443 -sd_ssl_port 9444 -sd_admin_name admin -sd_admin_password
secret -admin_user admin -admin_email "admin@example.com" -admin_password
secret -agent_key_size 2048 -agent_key_type rsa -agent_cert_subject "cn=tps\
agent\ cert" -ldap_host server -ldap_port 389 -bind_dn "cn=directory\ manager"
-bind_password secret -base_dn "o=pki-tps2" -db_name "server.example.com-pki-
tps2" -ca_hostname server.example.com -ca_port 9180 -ca_ssl_port 9443 -
tkc_hostname server.example.com -tkc_ssl_port 13443 -drm_hostname
server.example.com -drm_ssl_port 10443 -key_size 2048 -key_type rsa -
tps_subsystem_cert_subject_name "cn=tps\ subsystem\ cert,o=testca\ domain" -
tps_server_cert_subject_name "cn=tps\ client\ cert,o=testca\ domain" -
tps_audit_signing_cert_subject_name "cn=audit\ signing\ cert,o=testca\ domain" -
ldap_auth_host auth.example.com -ldap_auth_port 389 -ldap_auth_base_dn
"ou=tps,ou=People,dc=example,dc=com"
```

7.3. Cloning a Subsystem Silently



IMPORTANT

Only CA instances can be cloned using **pkisilent**. The other subsystem clones must be configured using the HTML-based configuration wizard.

When creating a new subsystem, there are options to set the type of keys to generate and to back up the keys to a PKCS #12 file. For cloning a subsystem, there are no key generation options. Instead, the parameters contain information pointing to the PKCS #12 file for the master subsystem and the URL for the subsystem to clone:

- -clone true (which sets that the new instance will be a clone)
- -clone_p12_file and -clone_p12_password, which gives the location of the PKCS #12 key file and the password to access it

Additionally, a clone must have some configuration in common with its master:

- The same security domain, set in the -sd_* parameters
- The same LDAP base DN and database name, set in the -ldap_* parameters (either the hostname or the port must be different, since the clone does require a separate Directory Server instance)
- The same issuing CA for its certificates, set in either the -ca_* parameters or possibly self-signed, for a CA

Aside from the differences in creating the subsystem certificates, the configuration for the clone (joining the security domain, creating the admin user, setting up the internal LDAP directories) is the same as with any other subsystem configuration.

For example:

```
pkisilent ConfigureCA -cs_hostname localhost -cs_port 9445 -subsystem_name "clone-
ca2" -client_certrdb_dir /tmp/ -client_certrdb_pwd password -preop_pin
sYY8er834FG9793fsef7et5 -sd_hostname "domain.example.com" -sd_admin_port 9445 -
sd_agent_port 9443 -sd_ssl_port 9444 -sd_admin_name admin -sd_admin_password
secret -admin_user admin -admin_email "admin@example.com" -admin_password secret
-clone true -clone_p12_file /export/backup.p12 -clone_p12_password secret -
master_instance_name pki-ca -ca_hostname server.example.com -ca_non_ssl_port
9180 -ca_ssl_port 9443 -ca_subsystem_cert_subject_name "cn=ca\ subsystem\
cert,o=testca\ domain" -ca_ocsp_cert_subject_name "cn=ocsp\ signing\
cert,o=testca\ domain" -ca_server_cert_subject_name "cn=ca\ client\ cert,o=testca\
domain" -ca_sign_cert_subject_name "cn=ca\ signing\ cert,o=testca\ domain" -
ca_audit_signing_cert_subject_name "cn=audit\ signing\ cert,o=testca\ domain"
```

7.4. Performing Silent Configuration Using an External CA

As described in [Section 4.1, “Requesting Subsystem Certificates from an External CA”](#), a CA outside of the security domain can be used to generate a subsystem's certificates. It is also possible to request and submit certificates issued by an external CA using **pkisilent**.

By default, the **pkisilent** command assumes that you will request a certificate from a CA within the security domain, and this CA is identified in the **-ca_hostname** and other **ca_** options. This assumes that the **-external** option is false.

To submit the subsystem certificate requests to an external CA, explicitly set the **-external** option to true. The generated certificate requests are exported to a file, and then can be submitted to the external CA. Once they are issued, files which contain the subsystem certificates and the CA certificate chain for the issuing external CA can be passed using the **pkisilent** command. This is set in four parameters:

- ▶ **-external**, which explicitly sets whether to use an external CA
- ▶ **-ext_csr_file**, which gives the path and name of the output file to which to write the certificate requests for the subsystem
- ▶ **-ext_ca_cert_file**, which gives the input file to use which contains the certificates issued by the external CA
- ▶ **-ext_ca_cert_file**, which gives the input file to use which contains the CA certificate chain for the external CA which issued the certificates

Whether it is performed through the HTML wizard or using **pkisilent**, submitting certificates to an external CA is a three-step process, two of them involving **pkisilent**:

1. In the first step, much of the preliminary information is configured for the instance. Along with the subsystem configuration settings, the subsystem's certificate requests are generated and written to the file specified in **-ext_csr_file**. These certificate requests must be submitted to the external CA.
For example (in real life, these options should be on a single line):

```
pkisilent ConfigureCA
  -cs_hostname server.example.com
  -cs_port 9445
  -subsystem_name "pki-ca2"
  -client_certrdb_dir /tmp/
  -client_certrdb_pwd password
  -preop_pin sYY8er834FG9793fsef7et5
  -domain_name "testca"
  -agent_name jsmith
  -agent_key_size 2048
  -agent_key_type rsa
  -agent_cert_subject "cn=ca\ agent\ cert"
  -ldap_host ldapserver.example.com
  -ldap_port 389
  -bind_dn "cn=directory\ manager"
  -bind_password password -base_dn "o=pki-ca2"
  -db_name "server.example.com-pki-ca2"
  -key_size 2048
  -key_type rsa
  -key_algorithm SHA512withRSA
  -token_name internal
  -token_pwd 242986083911
  -save_p12 true
  -backup_pwd password
  -backup_fname /export/backup.p12
  -ca_subsystem_cert_subject_name "cn=ca\ subsystem\ cert,o=testca\
domain"
  -ca_ocsp_cert_subject_name "cn=ocsp\ signing\ cert,o=testca\ domain"
  -ca_server_cert_subject_name "cn=ca\ client\ cert,o=testca\ domain"
  -ca_sign_cert_subject_name "cn=ca\ signing\ cert,o=testca\ domain"
  -ca_audit_signing_cert_subject_name "cn=audit\ signing\ cert,o=testca\
domain"
  -external true
  -ext_csr_file /tmp/cert.req
```

2. The certificate requests are submitted to the external CA, and the issued certificates are retrieved and saved to file.
3. The newly issued subsystem certificates are installed in the instance by referencing the saved certificate file in the **-ext_cert_file** parameter and the issuing CA's certificate chain in the **-ext_cert_chain_file** parameter.

For example (in real life, these options should be on a single line):

```
pkisilent ConfigureCA
  -cs_hostname server.example.com
  -cs_port 9445
  -subsystem_name "pki-ca2"
  -client_certrdb_dir /tmp/
  -client_certrdb_pwd password
  -preop_pin sYY8er834FG9793fsef7et5
  -domain_name "testca"
  -admin_user admin
  -admin_password secret
  -admin_email "admin@example.com"
  -agent_name jsmith
  -agent_key_size 2048
  -agent_key_type rsa
  -agent_cert_subject "cn=ca\ agent\ cert"
  -ldap_host ldapserver.example.com
  -ldap_port 389
  -bind_dn "cn=directory\ manager"
  -bind_password password
  -base_dn "o=pki-ca2"
  -db_name "server.example.com-pki-ca2"
  -key_size 2048
  -key_type rsa
  -key_algorithm SHA512withRSA
  -token_name internal
  -token_pwd 242986083911
  -save_p12 true
  -backup_pwd password
  -backup_fname /export/backup.p12
  -ca_subsystem_cert_subject_name "cn=ca\ subsystem\ cert,o=testca\
domain"
  -ca_ocsp_cert_subject_name "cn=ocsp\ signing\ cert,o=testca\ domain"
  -ca_server_cert_subject_name "cn=ca\ client\ cert,o=testca\ domain"
  -ca_sign_cert_subject_name "cn=ca\ signing\ cert,o=testca\ domain"
  -ca_audit_signing_cert_subject_name "cn=audit\ signing\ cert,o=testca\
domain"
  -external true
  -ext_cert_file /tmp/cert.cer
  -ext_cert_chain_file /tmp/cachain.cer
```

This is also when the final configuration to create the administrator user is performed.



NOTE

All of the previous parameters must be included the second time that **pkisilent** is invoked.

Chapter 8. Updating and Removing Subsystem Packages

Certificate System is installed using individual packages for each of its subsystems and its supporting systems, like Red Hat Directory Server and NSS. This makes the Certificate System modular, and each individual subsystem and its packages can be installed, updated, and removed independently.

8.1. Updating Certificate System Packages

There are many packages, listed in [Section 2.3, “Packages Installed on Red Hat Enterprise Linux”](#), installed with Certificate System for related applications and dependencies, not just the subsystem packages. For all supported platforms, individual Certificate System packages may be updated through the native package utility, **yum**.



NOTE

All Certificate System instances must be stopped before beginning any updates.

The recommended way to update packages is to use **yum** to manage the updates.

1. Stop all Certificate System instances.

```
service instance_ID stop
```

2. Log in as **root**.
3. Run **yum** for the package. For example:

```
yum update pki-java-tools-8.0.0-4.noarch
```

Or simply:

```
yum update pki-java-tools
```

4. Restart the Certificate System instances.

```
service instance_ID start
```

Alternatively, any updated RPMs can be downloaded and installed manually.

1. Stop all Certificate System instances.

```
service instance_ID stop
```

2. Log in as **root**.
3. Install the updated package.

```
rpm -Uvh package_name
```

For example:

```
rpm -Uvh pki-java-tools-8.0.0-4.noarch.rpm
```

4. Restart the Certificate System instances.

```
service instance_ID start
```

8.2. Uninstalling Certificate System Subsystems

It is possible to remove individual subsystem instances or to uninstall all packages associated with an entire subsystem. Instances and subsystems are installed and uninstalled individually. For example, it is possible to uninstall a DRM subsystem while leaving an installed and configured CA subsystem. It is also possible to remove a single CA instance while leaving other CA instances on the machine.

8.2.1. Removing a Subsystem Instance

Removing an instance requires specifying the instance directory and the instance name. This command removes all files associated with the instance (without removing the subsystem packages).

```
pkiremove -pki_instance_root=pki_instance_root -pki_instance_name=pki_instance_ID -force
```

The *pki_instance_root* is the directory path of the instance, such as **/var/lib**. The *pki_instance_name* is the instance name, such as **pki-ca**.



TIP

Use **-force** with **pkiremove** to remove the instance without prompting for confirmation.

Example 8.1. Removing a CA Instance

```
pkiremove -pki_instance_root=/var/lib -pki_instance_name=pki-ca1

PKI instance Deletion Utility ...

PKI instance Deletion Utility cleaning up instance ...

Stopping pki-ca1:
process already stopped

Removing dir /var/lib/pki-ca1
Removing file /var/log/pki-ca1-install.log
Removing file /etc/init.d/pki-ca1
Removing file /usr/share/applications/pki-ca1-config.desktop
Removing file /usr/bin/dtomcat5-pki-ca1
```

pkiremove removes the instance and any related files, such as the certificate databases, certificates, keys, and associated users. It does not uninstall the subsystem packages.

8.2.2. Removing Certificate System Subsystem Packages

A number of subsystem-related packages and dependencies are installed with Red Hat Certificate System; these are listed in [Section 2.3, “Packages Installed on Red Hat Enterprise Linux”](#). Removing a subsystem instance removes only the files and directories associated with that specific instance. It does not remove the actual installed packages that are used by that instance. Completely uninstalling Red Hat

Certificate System or one of its subsystems requires using package management tools, like **yum**, to remove each package individually.

To uninstall an individual Certificate System subsystem packages:

1. Remove all the associated subsystem instances using **pkiremove**. For example:

```
pkiremove -pki_instance_root=/var/lib -pki_instance_name=pki-ca
```

2. Run the uninstall utility. For example:

```
yum remove pki-subsystem_type
```

The subsystem type can be **ca**, **ra**, **drm**, **ocsp**, **tk**s, or **tp**s.

3. To remove other packages and dependencies, remove the packages specifically, using **yum**. The complete list of installed packages is at [Section 2.3, “Packages Installed on Red Hat Enterprise Linux”](#).

Chapter 9. Using Certificate System

Using any Certificate System has basic tasks such as editing the configuration file, starting and stopping the server instance and Console, opening web services, and locating logs. This is explained in more detail in the *Certificate System Administrator's Guide*.

9.1. Starting the Certificate System Console

The CA, DRM, OCSP, and TKS subsystems have a Java interface which can be accessed to perform administrative functions. For the DRM, OCSP, and TKS, this includes very basic tasks like configuring logging and managing users and groups. For the CA, this includes other configuration settings such as creating certificate profiles and configuring publishing.

The Console is opened by connecting to the subsystem instance over its SSL port using the **pkiconsole** command. This command has the format:

```
pkiconsole https://server.example.com:admin_port/subsystem_type
```

The *subsystem_type* can be **ca**, **kra**, **ocsp**, or **tk**s. For example, this opens the DRM console:

```
pkiconsole https://server.example.com:10445/kra
```

If DNS is properly configured, then an IPv4 or IPv6 address can be used to connect to the console. For example:

```
https://1.2.3.4:9445/ca  
https://[00:00:00:00:123:456:789:00:]:9445/ca
```

9.2. Starting, Stopping, and Restarting an Instance

The Certificate System subsystem instances can be stopped and started using system tools on Red Hat Enterprise Linux. For example:

```
service instance-name {start|stop|restart}
```

The instance name for default subsystem instances is usually **pki-instance-id**, such as **pki-ca**.

9.3. Starting the Subsystem Automatically

Red Hat Enterprise Linux 5.3 has a tool called **chkconfig** which manages the automatic startup and shutdown settings for each process on the server. This means that when a system reboots, some services can be automatically restarted. **chkconfig** also defines startup settings for different run levels of the server. **chkconfig** is explained more in the Red Hat Enterprise Linux documentation, such as [the Deployment Guide](#).

Certificate System subsystems can be managed by **chkconfig**, so this tool can set whether to restart subsystems automatically. By default, every Certificate System subsystem instance is turned off at every run level in the system, meaning instances must be started and stopped manually. This can be changed by resetting the configuration in **chkconfig** to **on**. For example, this automatically restarts Red Hat Directory Server, Administration Server, and the CA:

```
/sbin/chkconfig --level 2345 dirsrv-admin on
/sbin/chkconfig --level 2345 dirsrv on
/sbin/chkconfig --level 2345 pki-ca on
```

Make sure the subsystem is listed with the other services.

```
chkconfig --list | grep subsystem_name
```

To remove the subsystem from the start list, simply turn the level to **off**:

```
chkconfig --level 35 subsystem_name off
```

Red Hat Enterprise Linux also has a GUI console that can manage **chkconfig** settings.

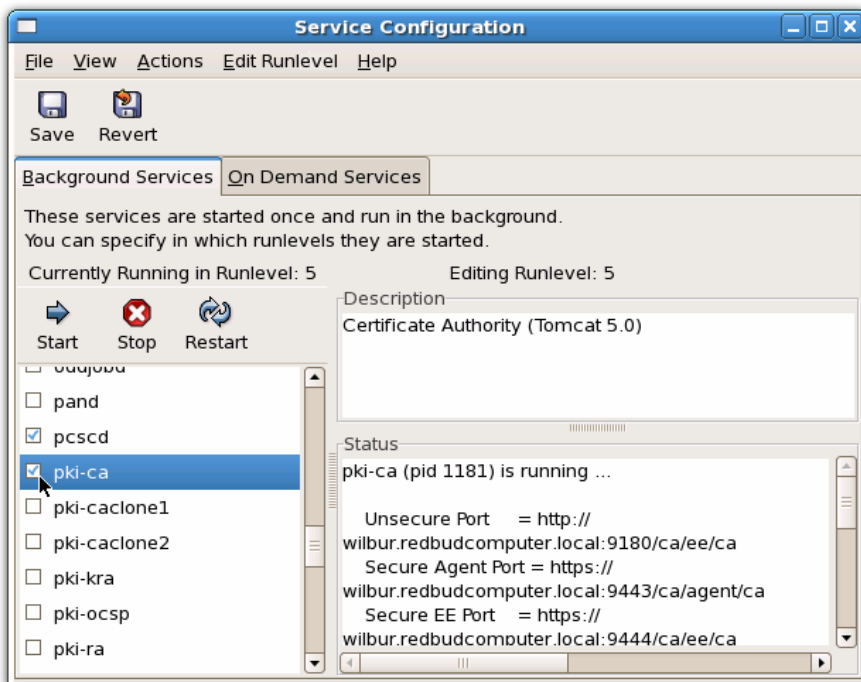


Figure 9.1. chkconfig Settings

The start order of services is extremely important, or the subsystems will not function. The Directory Server and Administration Server instances used by the subsystems must be running before the subsystems can be started, and their web services (Tomcat or Apache) must be running before the subsystems are started or their web services will not function.

The default Certificate System **chkconfig** settings set a start and stop priority for all of the subsystems and their dependent services so that they start and stop in the proper order, as listed in [Table 9.1, “Certificate System Processes and Their chkconfig Start Priority”](#). Processes with a low number for their start priority are started first, so Directory Server, Administration Server, and Tomcat are started before any of the subsystem instances. Likewise, processes with a low number for their shutdown priority are shut down first, so the subsystem processes are stopped before the processes they depend on are stopped.

Table 9.1. Certificate System Processes and Their chkconfig Start Priority

| Server | Process Name | Start Priority | Shutdown Priority |
|-----------------------|--------------|----------------|-------------------|
| Administration Server | dirsrv-admin | 21 | 79 |
| Directory Server | dirsrv | 21 | 79 |
| Tomcat Server | tomcat5 | 80 | 20 |
| CA | pki-ca | 81 | 19 |
| DRM | pki-kra | 82 | 18 |
| OCSP | pki-ocsp | 83 | 17 |
| TKS | pki-tks | 84 | 16 |
| Apache | httpd | 85 | 15 |
| RA | 86 | 14 | |
| TPS | pki-tps | 87 | 13 |

9.4. Finding the Subsystem Web Services Pages

The CA, RA, DRM, OCSP, TKS, and TPS subsystems have web services pages for agents, as well as potentially regular users and administrators. These web services can be accessed by opening the URL to the subsystem host over the subsystem's secure end user's port. For example, for the CA:

```
https://server.example.com:9444/ca/services
```



TIP

To get a complete list of all of the interfaces, URLs, and ports for a subsystem, check the service's status:

```
service instance-name status
```

The main web services page for each subsystem has a list of available services pages; these are summarized in [Table 9.2, “Default Web Services Pages”](#). To access any service specifically, access the appropriate port and append the appropriate directory to the URL. For example, to access the CA's end entities (regular users) web services:

```
https://server.example.com:9444/ca/ee/ca/
```

If DNS is properly configured, then an IPv4 or IPv6 address can be used to connect to the services pages. For example:

```
https://1.2.3.4:9444/ca/services
https://[00:00:00:00:123:456:789:00]:9444/ca/services
```



NOTE

Anyone can access the end user pages for a subsystem, but accessing agent or admin web services pages requires that an agent or administrator certificate be issued and installed in the web browser, or authentication to the web services fails.

Table 9.2. Default Web Services Pages

| Port | Used for SSL | Used for Client Authentication ^[a] | Web Services | Web Service Location |
|------------------------------------------|--------------|-----------------------------------------------|-----------------------------|------------------------------------------------|
| Certificate Manager | | | | |
| 9180 | No | | End Entities | ca/ee/ca/ |
| 9444 | Yes | No | End Entities | ca/ee/ca |
| 9443 | Yes | Yes | Agents | ca/agent/ca |
| 9445 | Yes | | Configuration | ca/admin/console/ config/login? pin=pin |
| 9445 | Yes | No | Services | ca/services |
| 9445 | Yes | No | Console | pkiconsole https://host:port/ca |
| Registration Manager | | | | |
| 12888 | No | | End Entities | ee/index.cgi |
| 12889 | Yes | Yes | Agents | agent/index.cgi |
| 12889 | Yes | Yes | Admin | admin/index.cgi |
| 12890 | Yes | | Configuration | ra/admin/console/ config/login? pin=pin |
| 12890 | Yes | | End Entities | ee/index.cgi |
| 12890 | Yes | | Services | index.cgi |
| Data Recovery Manager | | | | |
| 10180 | No | | End Entities ^[b] | kra/ee/kra/ |
| 10444 | Yes | No | End Entities ^[b] | kra/ee/kra |
| 10443 | Yes | Yes | Agents | kra/agent/kra |
| 10445 | Yes | | Configuration | kra/admin/console/ config/login? pin=pin |
| 10445 | Yes | No | Services | kra/services |
| 10445 | Yes | No | Console | pkiconsole https://host:port/kra |
| Online Certificate Status Manager | | | | |
| 11180 | No | | End Entities ^[c] | ocsp/ee/ocsp |
| 11444 | Yes | No | End Entities ^[c] | ocsp/ee/ocsp |
| 11443 | Yes | Yes | Agents | ocsp/agent/ocsp |
| 11445 | Yes | | Configuration | ocsp/admin/conso le/config/login? |

| | | | | |
|--------------------------------|-----|-----|---------------------------------------------------------|-------------------------------------------------|
| | | | | pin= <i>pin</i> |
| 11445 | Yes | No | Services | ocsp/services |
| 11445 | Yes | No | Console | pkiconsole https://host:port/ocsp |
| Token Key Service | | | | |
| 13180 | No | | End Entities ^[b] | tkes/ee/tks |
| 13444 | Yes | No | End Entities ^[b] | tkes/ee/tks |
| 13443 | Yes | Yes | Agents | tkes/agent/tks |
| 13445 | Yes | | Configuration | tkes/admin/console/config/login?pin= <i>pin</i> |
| 13445 | Yes | No | Services | tkes/services |
| 13445 | Yes | No | Console | pkiconsole https://host:port/tks |
| Token Processing System | | | | |
| 7888 | No | | Enterprise Security Client Phone Home | cgi-bin/home/index.cgi |
| 7890 | Yes | | Enterprise Security Client Phone Home | cgi-bin/home/index.cgi |
| 7888 | No | | Enterprise Security Client Security Officer Enrollment | cgi-bin/so/enroll.cgi |
| 7890 | Yes | Yes | Enterprise Security Client Security Officer Enrollment | cgi-bin/so/enroll.cgi |
| 7889 | Yes | Yes | Enterprise Security Client Security Officer Workstation | cgi-bin/sow/welcome.cgi |
| 7889 | Yes | Yes | Agents ^[d] | tus |
| 7889 | Yes | Yes | Admin ^[d] | tus? op=index_admin |
| 7889 | Yes | Yes | Operator ^[d] | tus? op=index_operator |
| 7890 | Yes | | Configuration | tps/admin/console/config/login?pin= <i>pin</i> |
| 7890 | Yes | | Services | index.cgi |

| | | | | |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----|---------|-----------------------------------------------------------------------|
| 9445 | Yes | No | Console | pkiconsole https://host:port/ca |
| <p>[a] Services with a client authentication value of No can be reconfigured to require client authentication. Services which do not have either a Yes or No value cannot be configured to use client authentication.</p> <p>[b] Although this subsystem type does have end entities ports and interfaces, these end-entity services are not accessible through a web browser, as other end-entity services are.</p> <p>[c] Although the OCSP does have end entities ports and interfaces, these end-entity services are not accessible through a web browser, as other end-entity services are. End user OCSP services are accessed by a client sending an OCSP request.</p> <p>[d] The agent, admin, and operator services are all accessed through the same web services page. Each role has a different tab on that page. The role-specific tab is visible to every user who is a member of that role.</p> | | | | |

9.5. Default File and Directory Locations for Certificate System

Certificate System servers consist of subsystems and instances.

Server subsystems are servers for a specific type of PKI function and are installed by the Certificate System RPMs. This general subsystem information is contained in non-relocatable, RPM-defined shared libraries, Java archive files, binaries, and templates. These are stored in a fixed location.



NOTE

There is an environment variable, **DONT_RUN_PKICREATE**, which stops the **pkicreate** script from running automatically after the subsystems are installed. This allows the default instances to be installed in user-defined installation directories, instead of the default locations in **var/lib**. To use custom directory locations, install the subsystems through the ISO image with this environment variable set to block the **pkicreate** script.

Server instances are somewhat relocatable and have user-specific default and customized forms and data.

When the Certificate System is first installed, one instance for each subsystem type is also installed. The default information such as the port numbers, instance name, and configuration file location for each subsystem (after being installed and going through the setup process) is listed in the following sections.

9.5.1. Default CA Instance Information

The default CA configuration is listed in [Table 9.3, “Default CA Instance Information”](#). Most of these values are unique to the default instance; the default certificates and some other settings are true for every CA instance.

Table 9.3. Default CA Instance Information

| Setting | Value |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Standard Port | 9180 |
| Agents Port | 9443 |
| End Users Port | 9444 |
| End-Entites Client Authentication Port | 9446 |
| Admin Port | 9445 |
| Tomcat Port | 9701 |
| Instance Name | pki-ca |
| Main Directory | /var/lib/pki-ca |
| Configuration Directory | /etc/pki-ca |
| Configuration File | /etc/pki-ca/CS.cfg /etc/pki-ca/password.conf |
| Subsystem Certificates | CA signing certificate OCSP signing certificate (for the CA's internal OCSP service) SSL server certificate Audit log signing certificate Subsystem certificate ^[a] |
| Security Databases | /var/lib/pki-ca/alias |
| Log Files | /var/log/pki-ca |
| Install Logs | /var/log/pki-ca-install.log |
| Process File | /var/run/pki-ca.pid |
| Profile Files | /var/lib/pki-ca/profiles/ca |
| Email Notification Templates | /var/lib/pki-ca/emails |
| Web Services Files | /var/lib/pki-ca/webapps - Agent services /var/lib/pki-ca/webapps.admin - Admin services /var/lib/pki-ca/webapps.ee - End user services |
| [a] The subsystem certificate is always issued by the security domain so that domain-level operations that require client authentication are based on this subsystem certificate. | |

9.5.2. Default RA Instance Information

The default RA configuration is listed in [Table 9.4, “Default RA Instance Information”](#). Most of these values are unique to the default instance; the default certificates and some other settings are true for every RA instance.

Table 9.4. Default RA Instance Information

| Setting | Value |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------|
| Standard Port (for End Users) | 12888 |
| SSL Port (for Agents and Administrators) | 12889 |
| SSL Port (for End Users) | 12890 |
| Instance Name | pki-ra |
| Main Directory | /var/lib/pki-ra |
| Configuration Directory | /etc/pki-ra |
| Configuration File | /etc/pki-ra/CS.cfg |
| | /etc/pki-ra/nss.conf |
| | /etc/pki-ra/password.conf |
| Subsystem Certificates | SSL server certificate |
| | Subsystem certificate ^[a] |
| Security Databases | /var/lib/pki-ra/alias |
| Log Files | /var/log/pki-ra |
| Install Logs | /var/log/pki-ra-install.log |
| Web Services Files | /var/lib/pki-ra/docroot |
| | /var/lib/pki-ra/lib |
| [a] The subsystem certificate is always issued by the security domain so that domain-level operations that require client authentication are based on this subsystem certificate. | |

9.5.3. Default DRM Instance Information

The default DRM configuration is listed in [Table 9.5, “Default KRA Instance Information”](#). Most of these values are unique to the default instance; the default certificates and some other settings are true for every DRM instance.

Table 9.5. Default KRA Instance Information

| Setting | Value |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| Standard Port | 10180 |
| End Users Secure Port | 10444 |
| Agents Port | 10443 |
| Admin Port | 10445 |
| Tomcat Port | 10701 |
| Instance Name | pki-kra |
| Main Directory | /var/lib/pki-kra |
| Configuration Directory | /etc/pki-kra |
| Configuration File | /etc/pki-kra/CS.cfg |
| | /etc/pki-kra/password.conf |
| Subsystem Certificates | Transport certificate Storage certificate SSL server certificate Audit log signing certificate Subsystem certificate ^[a] |
| Security Databases | /var/lib/pki-kra/alias |
| Log Files | /var/log/pki-kra |
| Install Logs | /var/log/pki-kra-install.log |
| Process File | /var/run/pki-kra.pid |
| Web Services Files | /var/lib/pki-kra/webapps - Agent services /var/lib/pki-kra/webapps.admin - Admin services |
| [a] The subsystem certificate is always issued by the security domain so that domain-level operations that require client authentication are based on this subsystem certificate. | |

9.5.4. Default OCSP Instance Information

The default OCSP configuration is listed in [Table 9.6, “Default OCSP Instance Information”](#). Most of these values are unique to the default instance; the default certificates and some other settings are true for every OCSP instance.

Table 9.6. Default OCSP Instance Information

| Setting | Value |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| Standard Port | 11180 |
| End Users Secure Port | 11444 |
| Agents Port | 11443 |
| Admin Port | 11445 |
| Tomcat Port | 11701 |
| Instance Name | pki-ocsp |
| Main Directory | /var/lib/pki-ocsp |
| Configuration Directory | /etc/pki-ocsp |
| Configuration File | /etc/pki-ocsp/CS.cfg /etc/pki-ocsp/password.conf |
| Subsystem Certificates | OCSP signing certificate SSL server certificate Audit log signing certificate Subsystem certificate ^[a] |
| Security Databases | /var/lib/pki-ocsp/alias |
| Log Files | /var/log/pki-ocsp |
| Install Logs | /var/log/pki-ocsp-install.log |
| Process File | /var/run/pki-ocspocsp.pid |
| Web Services Files | /var/lib/pki-ocsp/webapps - Agent services /var/lib/pki-ocsp/webapps.admin - Admin services |
| [a] The subsystem certificate is always issued by the security domain so that domain-level operations that require client authentication are based on this subsystem certificate. | |

9.5.5. Default TKS Instance Information

The default TKS configuration is listed in [Table 9.7, “Default TKS Instance Information”](#). Most of these values are unique to the default instance; the default certificates and some other settings are true for every TKS instance.

Table 9.7. Default TKS Instance Information

| Setting | Value |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|
| Standard Port | 13180 |
| End Users Secure Port | 13444 |
| Agents Port | 13443 |
| Admin Port | 13445 |
| Tomcat Port | 13701 |
| Instance Name | pki-tks |
| Main Directory | /var/lib/pki-tks |
| Configuration Directory | /etc/pki-tks |
| Configuration File | /etc/pki-tks/CS.cfg /etc/pki-tks/password.conf |
| Subsystem Certificates | SSL server certificate Audit log signing certificate Subsystem certificate ^[a] |
| Security Databases | /var/lib/pki-tks/alias |
| Log Files | /var/log/pki-tks |
| Install Logs | /var/log/pki-tks-install.log |
| Process File | /var/run/pki-tks.pid |
| [a] The subsystem certificate is always issued by the security domain so that domain-level operations that require client authentication are based on this subsystem certificate. | |

9.5.6. Default TPS Instance Information

The default TPS configuration is listed in [Table 9.8, “Default TPS Instance Information”](#). Most of these values are unique to the default instance; the default certificates and some other settings are true for every TPS instance.

Table 9.8. Default TPS Instance Information

| Setting | Value |
|------------------------------------------|------------------------------------------------------------------------------|
| Standard Port (for End Users) | 7888 |
| SSL Port (for Agents and Administrators) | 7889 |
| SSL Port (for End Users) | 7890 |
| Instance Name | pki-tps |
| Main Directory | /var/lib/pki-tps |
| Configuration Directory | /etc/pki-tps |
| Configuration File | /etc/pki-tps/CS.cfg /etc/pki-tps/nss.conf /etc/pki-tps/password.conf |
| Subsystem Certificates | SSL server certificate Subsystem certificate |
| Security Databases | /var/lib/pki-tps/alias |
| Log Files | /var/log/pki-tps |
| Install Logs | /var/log/pki-tps-install.log |
| Web Services Files | /var/lib/pki-tps/docroot /var/lib/pki-tps/cgi-bin /var/lib/pki-tps/lib |

9.5.7. Shared Certificate System Subsystem File Locations

There are some directories used by or common to all Certificate System subsystem instances for general server operations, listed in [Table 9.9, “Subsystem File Locations”](#).

Table 9.9. Subsystem File Locations

| Directory Location | Contents |
|------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>/var/lib/instance_name</code> | Contains the main instance directory, which is the location for user-specific default and customized configuration files, profiles, certificate databases, web files, and other files for the subsystem instance. |
| <code>/usr/share/java/pki</code> | Contains Java archive files shared by the Certificate System subsystems. Along with shared files for all subsystems, there are subsystem-specific files in subfolders: <div> <p><code>pki/ca/</code> (CA)</p> <p><code>pki/kra/</code> (DRM)</p> <p><code>pki/ocsp/</code> (OCSP)</p> <p><code>pki/tks/</code> (TKS)</p> </div> <i>Not used by the RA or TPS subsystems.</i> |
| <code>/usr/share/pki</code> | Contains common files and templates used to create Certificate System instances. Along with shared files for all subsystems, there are subsystem-specific files in subfolders: <div> <p><code>pki/ca/</code> (CA)</p> <p><code>pki/kra/</code> (DRM)</p> <p><code>pki/ocsp/</code> (OCSP)</p> <p><code>pki/ra/</code> (RA)</p> <p><code>pki/tks/</code> (TKS)</p> <p><code>pki/tps</code> (TPS)</p> </div> |
| <code>/usr/bin</code> | Contains the pkicreate and pkiremove instance configuration scripts and tools (Java, native, and security) shared by the Certificate System subsystems. |
| <code>/var/lib/tomcat5/common/lib</code> | Contains Java archive files shared by local Tomcat web applications and shared by the Certificate System subsystems. <i>Not used by the TPS or RA subsystems.</i> |
| <code>/var/lib/tomcat5/server/lib</code> | Contains Java archive files used by the local Tomcat web server and shared by the Certificate System subsystems. <i>Not used by the TPS or RA subsystems.</i> |
| <code>/usr/lib/httpd/modules</code> <code>/usr/lib64/httpd/modules</code> | Contains Apache modules shared by TPS and RA subsystems. <i>Not used by the CA, DRM, OCSP, or TKS subsystems.</i> |

/usr/lib/mozldap
/usr/lib64/mozldap

Mozilla LDAP SDK tools shared by TPS and RA subsystems. *Not used by the CA, DRM, OCSP, or TKS subsystems.*

Index

A

accelerators, [Hardware Cryptographic Accelerators](#)

additional installation options

- changing hashing algorithm for subsystem keys, [Changing the Hashing Algorithm Used for Subsystem Keys](#)

C

certificate

- lifecycle, [Subsystems for Managing Certificates](#)

Certificate System

- starting and stopping, [Starting, Stopping, and Restarting an Instance](#)

E

external tokens

- defined, [External Tokens](#)
- installing, [Installing External Tokens and Unsupported HSM](#)

H

hardware accelerators, [Hardware Cryptographic Accelerators](#)

hardware tokens, [Setting up Tokens for Storing Certificate System Subsystem Keys and Certificates](#)

- See external tokens, [External Tokens](#)

hashing algorithms

- changing, [Changing the Hashing Algorithm Used for Subsystem Keys](#)

I

installation, [Installation and Configuration](#)

- planning, [Planning the Installation](#)

installing external hardware tokens, [Installing External Tokens and Unsupported HSM](#)

internal tokens, [Internal Tokens](#)

K

keys

- changing hashing algorithm, [Changing the Hashing Algorithm Used for Subsystem Keys](#)

P

PKCS #11 support, [Setting up Tokens for Storing Certificate System Subsystem Keys and Certificates](#), [External Tokens](#)

planning installation, [Planning the Installation](#)

S

security domains, [Subsystems for Managing Certificates](#)

subsystems for certificates, [Subsystems for Managing Certificates](#)

- Certificate Manager, [Certificate Manager](#)
- Data Recovery Manager, [Data Recovery Manager](#)
- Online Certificate Status Manager, [Online Certificate Status Manager](#)
- overview, [Subsystems for Managing Certificates](#)
- Registration Authority, [Registration Authority](#)

subsystems for tokens, [Subsystems for Managing Tokens](#)

- Enterprise Security Client, [Enterprise Security Client](#)
- overview, [Subsystems for Managing Tokens](#)
- Token Key Service, [Token Key Service](#)
- Token Processing System, [Token Processing System](#)

T

tokens

- defined, [Setting up Tokens for Storing Certificate System Subsystem Keys and Certificates](#), [Types of Hardware Tokens](#)
- external, [Setting up Tokens for Storing Certificate System Subsystem Keys and Certificates](#), [External Tokens](#)
- internal, [Internal Tokens](#)
- viewing which tokens are installed, [Viewing Tokens](#)